Synthesising End-to-End Security Schemes through Endorsement Intermediaries

A thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

Charles Thevathayan BSc (Hons), MAppSc
School of Computer Science and Information Technology,
College of Science, Engineering and Health (SEH),
RMIT University
Melbourne, Victoria, Australia

February, 2013

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

P. Charles Thevathayan

21 January 2013

Acknowledgements

I would like to express my sincere gratitude to both my supervisors Associate Professor Peter Bertok and Associate Professor James Harland for their patience and guidance. I would also like to specially thank my former second supervisor Associate Professor George Fernandez for all his encouragement and support. Without their help and guidance this work would not be possible. I would also like to thank Professor Zahir Tari and my external examiners for their comments and suggestions. I am thankful to the school of CSIT for giving me this opportunity. I greatly appreciate the proofreaders Ms Alison Caddick and Ms Nicole Preiss for their valuable comments and suggestions. A very special thanks goes to my wife and sons who put up with my irregular hours of work over many years. Above all I would like to thank my Lord and Saviour Jesus for His mercy and grace.

Credits

Portions of this thesis were published or presented in the following conferences:

Craig Pearce, Peter Bertok and Charles Thevathayan. *A Protocol for Secrecy and Authentication within Proxy-Based SPKI/SDSI Mobile Networks* in *Asia Pacific Information Technology Security Conference* 2004: Gold Coast, Australia.

Charles Thevathayan, James Harland and Peter Bertok. *Synthesising End-to-End Security Protocols for E-Commerce* in *Australasian Information Security Conference* 2013: Adelaide, Australia (presentation only).

Charles Thevathayan, James Harland and Peter Bertok. *Endorsement Trust Model* in *12th International Conference on Trust, Security and Privacy in Computing and Communications* 2013: Melbourne Australia.

Table of Contents

Credits ..... *iii*
Table of Contents ..... *iv*
Table of Definitions ..... *xi*
List of Figures ..... *xiii*
List of Tables ..... *xv*
Abstract ..... *xvii*
Chapter 1: Introduction *1*
1.1 E-Commerce Security Protocols 3
1.1.1 Standard E-Commerce Security Protocols 5
1.1.1.1 The SSL Protocol (now known as TLS) 5
1.1.1.2 The SET Protocol 6
1.1.2 Past Research and the need for a Holistic Approach to Synthesis 7
1.1.3 Difficulties Creating Security Protocols for E-Commerce 7
1.2 Challenges Facing Synthesis of E-Commerce Protocols Incorporating Security and Trust 8
1.3 Research Questions 11
1.3.1 Main Research Questions Pursued in this Thesis 11
1.4 The Approach Proposed in this Thesis 12
1.5 The Benefits/Rationale of the Proposed Framework 14
1.6 Contributions 17
1.7 Thesis Structure 20
Chapter 2: Background and Mathematical Foundations *21*
2.1 Security Protocols 21
2.1.1 Main Elements of a Security Protocol 21
2.1.2 Security Protocol Assumptions 23
2.1.3 Example of a Security Protocol: Key Exchange Protocol 23
2.2 Protocol Verification 23
2.2.1 Theorem Proving 24
2.2.1.1 BAN Logic 24
2.2.1.2 Strand Spaces 25
2.3 Previous Work on Protocol Synthesis 27
2.3.1 Search-Based Synthesis Techniques 28
2.3.1.1 Automatic Protocol Generation and Selection 28
2.3.1.2 Backward Search from Protocol Goals 28
2.3.1.3 Security Protocol Generation through Meta-heuristic Search 29
2.3.2 Compositional Approach to Synthesis 29
2.3.2.1 Strand Spaces Based Composition 29

2.3.2.2 Secure Protocol Composition Logic 30
2.3.3 Semantics Based Protocol Synthesis 34
2.3.4 Overall Evaluation of Past Synthesis Techniques 34
2.4 Security Protocol Attacks and Prevention Techniques 35
2.4.1 Requirements for Security Protocols 35
2.4.2 Common Attacks and Techniques for Prevention 36
2.4.2.1 Type-Flaw Attack 36
2.4.2.2 Replay Attack 37
2.4.3 Protocol Design Guidelines To Prevent Common Attacks 38
2.4.3.1 Principle of Full Information 38
2.4.3.2 Explicitly Stating the Reasons for Cryptographic Components 39
2.4.4 Summary of Common Protocol Attacks 40
2.5 Accountability in E-Commerce Protocols 41
2.6 Performance Trade-offs in Security Protocols 42
2.7 Application of Past Logics/Techniques in this Thesis 44
Chapter 3: Protocol Generation using Proven Schemes (PGPS) *45*
3.1 What is PGPS (Protocol Generation using Proven Schemes)? 45
3.2 Motivation: End-to-End Security for E-Commerce Protocols 46
3.2.1 SET Protocol Overview 47
3.2.2 SET Purchase Protocol Analysis 49
3.2.3 The Proposed Solution 50
3.2.4 SET Purchase Protocol Specifications in PGPS 51
3.2.5 Equivalent SET Purchase Protocol Composed using PGPS Technique 52
3.2.6 PGPS Overview 54
3.2.6.1 Inputs to PGPS 54
3.2.6.2 PGPS Business Constraints 54
3.2.6.3 PGPS End-to-End Security Requirements 55
3.2.7 PGPS: Research Contribution 57
3.3 Statement of the Problem 58
3.3.1 Research Questions 59
3.4 Outline of the Solution 60
3.4.1 Enforcing Basic Properties 60
3.4.2 Emergent Behaviours of Composite Properties 61
3.4.3 PGPS Security Properties and Security Levels 61
3.4.4 Aggregation of Security Properties 63
3.4.5 Methodology for Creating Fine-grained Schemes 63
3.4.6 Extending Schemes to Multiple Recipients 63
3.5 The Proposed Model (PGPS) 64

3.5.1 Rationale and Overview of PGPS Model 64
3.5.2 Model Elements 65
3.5.2.1 A Motivating Example: Interaction between Semantics and Security 65
3.5.2.2 Specifying Security Requirements 65
3.5.2.3 Expressing Fine-grained Levels of Security 66
3.5.2.4 Cost of Security Protocols 66
3.5.2.5 Schemes Enforcing Secrecy 66
3.5.2.6 Schemes Enforcing Correspondence Properties 67
3.5.2.7 Scheme Generators 67
3.6 PGPS Details 69
3.6.1 PGPS Techniques for Preventing Common Attacks 69
3.6.1.1 Preventing Replay Attacks 70
3.6.1.2 Preventing Type-Flaw Attacks 70
3.6.2 Security Mechanisms to Meet Semantics of Properties 71
3.6.2.1 Data Authentication 71
3.6.2.2 Entity Authentication 71
3.6.2.3 Data Integrity 72
3.6.2.4 Data Recency 72
3.6.2.5 Partial Receiver Non-Repudiation 72
3.6.2.6 Secrecy 73
3.6.3 Schemes for Basic Properties and Proofs 74
3.6.3.1 Proof System 75
3.6.4 Composite Properties and Schemes Enforcing Them 81
3.6.4.1 Standardised Emergent Behaviours 81
3.6.4.2 Techniques for Enforcing Non-Interference 82
3.6.4.3 Reducing Overheads in Combined Schemes 82
3.6.4.4 Forming Composite Schemes 83
3.6.4.5 Composed Schemes 85
3.6.5 Schemes for Multiple Recipients and Their Proofs 91
3.6.5.1 Multiple Recipients Security Scheme Generation Algorithm 91
3.6.5.2 Multiple Recipients Security Schemes: Proofs 94
3.6.6 PGPS Performance 98
3.6.6.1 Analysis of PGPS Protocol Generation Algorithm 98
3.6.6.2 Performance of the Protocol Generation Algorithm 99
3.6.6.3 Security Overheads of PGPS Generated Schemes 101
3.6.6.4 Computational Cost 101
3.7 Applicability: Dynamic Generation of Security Protocols 104
3.7.1 Automated Synthesis (SYN) 104

3.7.1.1 Model Elements and the Basis for Synthesis 105
3.7.2 Performance of PGPS 110
3.8 Discussion 111
3.8.1 Security Requirements using Fine-Grained Security Levels 111
3.8.2 Synthesising Protocols Using Proven Schemes 112
3.8.3 Extending Security Schemes for E-Commerce 112
3.8.4 Cost Estimation Based on underlying Cryptographic elements 113
3.8.5 Future Work 113
3.9 Conclusion 114
Chapter 4: Promoting Trust via Endorsement Intermediaries *115*
4.1 Introduction 115
4.1.1 Literature Review 116
4.1.1.1 Trust Characteristics 117
4.1.1.2 Trust Evolution 117
4.1.1.3 The Role of Trust Service Providers 117
4.1.1.4 Economic Incentives for Trust Service Providers 118
4.1.1.5 Need for Endorsements 118
4.1.1.6 Trusted Paths 119
4.1.1.7 Direct and Indirect Trust Propagation 119
4.1.1.8 Transitive Trust 119
4.1.1.9 Summary of Literature Review and Evaluation 121
4.1.2 Main Contribution 122
4.2 Statement of the Problem 123
4.2.1 Research Questions 123
4.3 Outline of the Solution 124
4.4 Endorsement-Trust Model and Building Blocks 126
4.4.1 Hierarchical Category Specific Endorsement Intermediaries 127
4.4.2 Endorsement-Trust Relationships 128
4.4.3 Hierarchical Category Specific Intermediary Network 130
4.4.4 Endorsement Processes 132
4.4.5 Entity and Domain Trust Policies 133
4.4.6 Path Trust (PT) and Path Endorsement Trust (PET) 134
4.4.7 Trusted Path Selection Criteria 135
4.4.8 Objective Criteria for Selecting Trusted Path 136
4.4.9 Setting Maximum Transitive-Depth 137
4.4.10 Trust Transfer and Trust Evolution 137
4.4.10.1 Network Initialisation 137

4.4.10.2 Trust Transfer Policies 137
4.4.10.3 Trust Evolution Policies 138
4.5 Internal Representation and Algorithms 139
4.5.1 Design Techniques to Address PTEI Implementation Goals 139
4.5.2 Internal Representation using Domain Trees and Sub-trees 140
4.5.2.1 Worst Case Memory Requirement Analysis 140
4.5.2.2 Minimising Search and Retrieval Times 141
4.5.3 PTEI Algorithms, Policies and Schemes 141
4.5.3.1 Features of PTEI Algorithm 141
4.5.3.2 Algorithm for Generating and Storing Domain Sub-Tree 142
4.5.3.3 Algorithm for Trusted Path Retrieval 143
4.5.3.4 Efficient Trusted Path Retrieval Using Hierarchical Domains 144
4.6 Experimental Results & Comparison with Other Work 146
4.6.1 Trust Evolution and Trust Transfer Parameters 147
4.6.2 Impact of Trust Coupling on Performance 149
4.6.3 Path Selection 151
4.6.4 Trusted Path Retrieval Time 151
4.6.5 Comparison with Existing Transitive Trust Models 153
4.6.6 Comparison with Other Trust Models 154
4.7 Discussion 155
4.7.1 Institutional Trust 155
4.7.2 Self-Regulating Trust Networks 156
4.7.3 Response Time for Trusted Path Generation 156
4.7.4 Security and Cost for Multiple Endorsements 157
4.7.5 Future Work 157
4.8 Conclusion 158
Chapter 5: Security Schemes for Endorsement Services (SSES) *159*
5.1 Introduction 159
5.1.1 Main Contribution 160
5.2 Statement of the Problem 161
5.2.1 Research Questions 161
5.3 Outline of the Solution 162
5.3.1 Deriving End-to-End Security Schemes 162
5.3.2 Schemes Devised for Endorsement Security 163
5.3.3 Technique for Enforcing Path Security 163
5.3.4 Enforcing Minimum Security Levels along a Message Path 163
5.3.5 Revised Schemes with Explicit Cryptographic Evidence 164
5.4 Literature Review 165

5.4.1 Security Threats of Content Transformation Intermediaries 165
5.4.2 Protocols for Intermediary Communication 165
5.4.3 Need for Fine-grained End-to-End Schemes 166
5.4.4 Accountability in E-commerce 166
5.4.4.1 Accountability Delegation 167
5.4.5 Summary and Evaluation 167
5.5 SSES Model Elements 168
5.5.1 Comparison with Interleaved Schemes Devised in Chapter 3 168
5.5.2 Endorsement Chain along Trusted Path 169
5.5.3 End-to-End Security Properties 170
5.5.4 Example of Server Signed Trusted Path 171
5.5.5 Enforcing Accountability through Proof Obligations 171
5.5.6 End-to-End Schemes for Data and Endorsement-Chain 173
5.5.7 Trust-Relationship and End-to-End Security 174
5.6 SSES Model Details 175
5.6.1 End-to-End Security Properties and Proof Obligations 175
5.6.1.1 End-to-End Authentication Property 175
5.6.1.2 End-to-End Time-bound Property 175
5.6.1.3 End-to-End Receiver Non-Repudiation Property and Scheme 175
5.6.1.4 End-to-End Data Integrity Property 175
5.6.1.5 End-to-End Secrecy Property 176
5.6.2 Discharging Proof Obligations 176
5.6.2.1 Assumptions 178
5.6.2.2 Postulates 179
5.6.2.3 Discharging Destination Proof Obligations 180
5.6.2.4 Discharging Source Proof Obligations 181
5.6.2.5 Attestable Evidence for Enforcing Security Properties 183
5.6.2.6 Discharging Multiple Proof Obligations 186
5.6.3 Role of Proof Obligations 187
5.6.4 Example: End-to-End Schemes Derived from Proof Obligations 188
5.6.4.1 End-To-End Authentication Scheme 188
5.6.4.2 End-to-End Data Integrity Scheme 189
5.6.4.3 End-to-End Time-Bound Scheme 189
5.6.4.4 End-to-End Receiver Non-Repudiation Scheme 190
5.6.5 Example: Composed End-to-End Schemes 191
5.6.5.1 End-to-End Scheme Combining Authentication and Time-Bound 191
5.6.5.2 End-to-End Scheme Combining Authentication and Non-Repudiation 191
5.6.5.3 End-to-End Scheme Combining Non-Repudiation and Time-Bound 192

5.6.5.4 End-To-End Scheme Combining Authentication and Time-Bound 192
5.6.5.5 Summary of Fine-grained End-to-End Security Schemes..... 192
5.6.6 Reducing the Cost of End-To-End Security Schemes..... 194
5.6.6.1 Variable Security Level for Reduced Overheads 194
5.6.6.2 Space Efficient Encoding of Variable Security Levels in Trusted Path..... 196
5.7 Discussion 197
5.7.1 Security Schemes Incorporating Intermediary Endorsements 197
5.7.2 Synthesising End-to-End Schemes from Proof Obligations 197
5.7.3 Mechanism for Enforcing End-to-End Schemes..... 198
5.7.4 Future Work 198
5.8 Conclusion..... 199
Chapter 6: Conclusion and Future Work..... *200*
6.1 Research Contributions 201
6.2 Future Work 204
Glossary *205*

Table of Definitions

DEFINITION 1.1 END-TO-END SECURITY *4*
DEFINITION 1.2 FINE-GRAINED SECURITY PROPERTIES *5*
TERMINOLOGY 2.1 COMMON TERMS IN SPCL *31*
DEFINITION 2.1 PREDICATE FORMULAS *31*
TERMINOLOGY 3.1 EMERGENT BEHAVIOUR *60*
TERMINOLOGY 3.2 CORRESPONDENCE PROPERTY *61*
TERMINOLOGY 3.3 SECURITY LEVEL *61*
TERMINOLOGY 3.4 DATA ELEMENTS AND DATA *61*
TERMINOLOGY 3.5 SECRECY GROUP *63*
DEFINITION 3.1 MESSAGE TERM *68*
TERMINOLOGY 3.6: NUMBER OF CYCLES IN A PROTOCOL SCHEME *71*
DEFINITION 3.2: PGPS PRIMITIVE ACTIONS *75*
DEFINITION 3.3: PGPS ASSERTIONS *75*
DEFINITION 3.4 PGPS INFERENCE RULES *76*
TERMINOLOGY 3.7: MODEL ELEMENTS *105*
TERMINOLOGY 3.8: DEPENDENCY CONSTRAINTS (DC) *107*
TERMINOLOGY 3.9:VALID MESSAGE RECEIPT (VMR) *107*
TERMINOLOGY 3.10: VALID MESSAGE DESPATCH (VMD) *107*
TERMINOLOGY 3.11: VALID MESSAGE RECEIVER GROUP (VMRG) *108*
TERMINOLOGY 4.1 COMMON TERMS *124*
DEFINITION 4.1 ENDORSEMENT TRUST RELATIONSHIP *129*
DEFINITION 4.2 TRUST COUPLING *130*
DEFINITION 4.3 ENDORSEMENT CAPABILITY (ECAP) *131*
DEFINITION 4.4 TRUST NETWORK (TN) *131*
DEFINITION 4.5 ENDORSEMENT PROCESS (EP) *132*
DEFINITION 4.6 ENTITY TRUST POLICY (ETP) *133*
DEFINITION 4.7 DOMAIN TRUST POLICY (DTP) *133*
DEFINITION 4.8 PATH TRUST *134*
DEFINITION 4.9 PATH ENDORSEMENT TRUST *135*
DEFINITION 4.10 OBJECTIVE CRITERIA FOR PATH SELECTION *136*
DEFINITION 4.11 REPUTATION *138*
DEFINITION 4.12 TRUST SPREAD *147*
DEFINITION 5.1 INTERMEDIARY ENDORSEMENT FUNCTION (F) *168*
TERMINOLOGY 5.1 TRUST SERVER *169*
TERMINOLOGY 5.2 PROOF OBLIGATION *172*
TERMINOLOGY 5.3 PRIMITIVE FORMULAS AND SEMANTICS *176*

DEFINITION 5.2 DESTINATION PROOF OBLIGATIONS *177*
DEFINITION 5.3 SOURCE PROOF OBLIGATIONS *177*

List of Figures

Figure 1.1 Hierarchy of Protocol Layers 6
Figure 1.2 Entities in SET Protocol 6
Figure 2.1 Strand Spaces Representation of Simple Protocol 26
Figure 3.1 Purchase Protocol Specification Made up of Data and Their Security Properties 51
Figure 3.2 Standard Set and PGPS Generated Protocol Steps 53
Figure 3.3 Lattice Model for PGIP Security Levels 62
Figure 3.4 Messages using Different Combinations of Security Properties 65
Figure 3.5 Mechanism for Authentication 71
Figure 3.6 Mechanism for Entity Authentication 71
Figure 3.7 Mechanism for Data Integrity 72
Figure 3.8 Mechanism for Data Recency 72
Figure 3.9 Mechanism for Receiver Non-Repudiation 73
Figure 3.10 Refined Schemes 84
Figure 3.11 Merging Security Schemes 85
Figure 3.12 Strand Spaces Bundle for Authentication Scheme 95
Figure 3.13 Strand Spaces Bundle for the Non-Repudiation Scheme 96
Figure 3.14 Strand Spaces Bundle for Recency Scheme 98
Figure 3.15 Protocol Generation Time for Varying Number of Recipients 100
Figure 3.16 Number of Protocol Terms for Varying Number of Recipients 100
Figure 3.17 Two Different Message Exchanges Producing the Same Outcomes 101
Figure 3.18 Specifying Dependenies between Knowledge-Elements 106
Figure 3.19 Dependencies between Knowledge Elements 109
Figure 3.20 Initial and Final Requirements and Dependency Constraints 109
Figure 3.21 Protocol Specifications at Message Level 110
Figure 4.1. PTEI Model Overview 126
Figure 4.2 Types of Endorsement Relationships 129
Figure 4.3 Network Showing Endorsement Paths and Category Specific Trust Relationships 132
Figure 4.4 Trusted Paths using Different Criteria 134
Figure 4.5 Trusted Path between Entity A and all others for the Network in Figure 4.3 135
Figure 4.6 Trust Network where Reputation between Sender A and Receiver B is 9. 138
Figure 4.7 Trust Degradation along the Trusted Path from E's Trust Breach 139
Figure 4.8 Organisation of Domain Trust Tree 140
Figure 4.9 Root of Sub-Tree Labelled with Trustworthiness and Transitive Depth for Each Entity 141
Figure 4.10 Hierarchical Clustering of Domains 145
Figure 4.11 Trust Spread with Varying Trust Transfer Threshold 148
Figure 4.12 Trust Spread through Trust Evolution Policies for Varying Reliability 148

Figure 4.13 Elapsed Time for Low, Medium and High Coupling (LNHMTD5Res) *150*
Figure 4.14 Percentage Entities Finding Trusted Paths for Varying Criteria *151*
Figure 4.15 Retrieval Time for Varying Number of Parameters *152*
Figure 4.16 Retrieval Time for Varying Number of Recipients *152*
Figure 4.17 Comparison of Trust Attenuation Rate for Endorsement and Transitive Trust *153*
Figure 5.1 Intermediary Messages with Fixed and Variable Parts *169*
Figure 5.2 Trusted Path from Message Originator A to Recipients C and D *171*
Figure 5.3 Destination and Source Proof Obligations *172*
Figure 5.4 Evidences Required to Delegate/Transfer/Discharge Proof Obligations *173*
Figure 5.5 Verifiable Basis Path for Intermediary Generated Responses *174*
Figure 5.6 End-to-End Non-Repudiation through delegation *180*
Figure 5.7 End-to-End Non-Repudiation through Delegation *181*
Figure 5.8 Successor Entities/Intermediaries along Trusted Path *182*
Figure 5.9 End-to-End Scheme for Authentication *189*
Figure 5.10 End-to-End Scheme for Data Integrity *189*
Figure 5.11 End-to-End Scheme for Time-Bound Property *189*
Figure 5.12 End-to-End Scheme for Non-Repudiation *190*
Figure 5.13 End-to-End Scheme for Authentication and Time-Bound Properties *191*
Figure 5.14 End-to-End Scheme for Authentication and Non-Repudiation *191*
Figure 5.15 End-to-End Scheme for Non-Repudiation and Time-Bound Properties *192*
Figure 5.16 End-to-End Scheme for Authentication, Non-Repudiation and Time-Bound properties *192*
Figure 5.17 Minimum Security-Level Based on Fine-Grained Security Requirements *195*

List of Tables

Table 1.1 Purpose of e-Commerce Security Properties 4
Table 1.2 Layers in the Framework 13
Table 2.1 Semantics of BAN Primitives 25
Table 2.2 Usage of Logics Presented in Subsequent Chapters 44
Table 3.1 Data Elements and their Security Requirements 49
Table 3.2 Grouping and Sequencing Requirements 50
Table 3.3 Entity Labels, Data Elements and Security Requirements 51
Table 3.4 Data Exchange Details 51
Table 3.5 PGPS Scheme Generated with SET Purchase Protocol Specifications 53
Table 3.6 E-Commerce Problems and PGPS Solutions 56
Table 3.7 Basic Correspondence Properties 62
Table 3.8 Non-correspondence Properties 62
Table 3.9 Symbols Used and Sample Values 68
Table 3.10 Summary of Schemes/Mechanisms for Basic Properties 74
Table 3.11 Protocol Axioms 76
Table 3.12 Assumptions and Invariants 77
Table 3.13 Emergent Behaviours Specifying Order of Enforcement for Security Properties 82
Table 3.14 Invariants Violated by Various Security Properties 83
Table 3.15 Composed Scheme for Authentication and Integrity 86
Table 3.16 Composed Scheme for Authentication and Receiver Non-Repudiation 86
Table 3.17 Composed Scheme for Authentication and Recency 86
Table 3.18 Composed Scheme for Integrity and Receiver Non-Repudiation 87
Table 3.19 Composed Scheme for Integrity and Recency 87
Table 3.20 Composed Scheme for Recency and Receiver Non-Repudiation 88
Table 3.21 Composed Scheme for Authentication, Integrity and Receiver Non-Repudiation 88
Table 3.22 Composed Scheme for Authentication, Receiver Non-Repudiation and Recency 89
Table 3.23 Composed Scheme for Authentication, Integrity and Recency 89
Table 3.24 Composed Scheme for Integrity, Receiver Non-Repudiation and Recency 90
Table 3.25 Composed Scheme for Authentication, Receiver Non-Repudiation and Recency 90
Table 3.26 Schemes Generated from A to [B,C] using Different Combination of Properties 93
Table 3.27 Capabilities of Adversary 94
Table 3.28 The Number of New Terms introduced for N^{th} Recipient and the Total Number of Terms 99
Table 3.29 Functions used for Describing Protocol Costs 102
Table 3.30 Computational Cost for Multiple Recipient Protocols 103
Table 3.31 Number of Headers Used for Security Levels 103
Table 3.32 Bandwidth for Multiple Recipient Protocols 104

Table 4.1 Extent of Trust Relationship (Category C1) *134*
Table 4.2 Specifying Objective Criteria for Path Selection *136*
Table 4.3 Recipients with Valid Trusted Paths *150*
Table 4.4 Comparing PTEI with other Trust Models *154*
Table 5.1 End-to-End Security Properties *170*
Table 5.2 Security Properties and their Indices *171*
Table 5.3 End-to-End Schemes for Authentication, Non-Repudiation and Time-Bound Properties *193*

Abstract

Web services, cloud computing and e-commerce require interaction protocols that can meet the functional and non-functional requirements of interacting entities. Composing secure interaction protocols dynamically continue to pose a number of challenges, such as lack of standard notations for expressing security requirements, difficulty in incorporating them into business processes, and the difficulty involved in enforcing them. Trust too plays a vital role as any interaction between unknown entities may require finding common trusted intermediaries. If such intermediaries are to be selected automatically trust relationships must be modelled explicitly reflecting past experiences. Such trust relationships must be made category specific as an intermediary trusted for medical transactions may not be trusted for finance related ones.

Securing messages sent through one or more intermediaries require devising schemes that provide end-to-end security guarantees. In the past, e-commerce protocols such as SET were created to provide such guarantees taking into consideration the underlying business processes, trust relationships and security requirements. However, such protocols assumed a standard configuration with fixed requirements and static trust relationships. Furthermore, complex hand crafted protocols proved difficult to model check. This thesis addresses the end-to-end problems in an open dynamic setting where trust relationships evolve, and requirements of interacting entities vary. Trust relationships too are assumed to vary reflecting past transactions and their categories. Interacting entities are allowed to specify trust, security and business related requirements for data elements exchanged. To reduce the need for model checking a structured synthesis approach was pursued.

Before interaction protocols can be synthesised at runtime based on underlying business, trust and security requirements, a number of research questions must be addressed. Firstly, to meet end-to-end security requirements, the security level along the message path must be made to reflect the requirements of all interacting parties. Thus, security schemes enforcing them must be created dynamically. Secondly, if entities are to interact with unknown entities using different types of messages at runtime, a form of institutional trust framework which provides category specific endorsements is necessary. Such a framework requires both direct and indirect endorsers as direct endorsers known to all parties may not exist. Thirdly, if all such endorsers are to be held liable, a technique must be devised to make them accountable.

The thesis proposes a number of solutions to address the research problems. End-to-end security requirements were arrived by aggregating security requirements of all interacting parties. These requirements were enforced by interleaving and composing basic schemes derived from challenge-response mechanisms. The trust policies were devised to allow evolution and establishments of trust relationships. The institutional trust promoting mechanism devised allowed all vital data to be

endorsed by authorised category specific intermediaries. Intermediaries were made accountable for their endorsements by being required to discharge or transfer proof obligations placed on them, using explicit cryptographic evidence.

The techniques devised for aggregating and enforcing security requirements allow creation of end-to-end security schemes at runtime. The novel interleaving technique devised allows creation of provably secure multiparty schemes for any number of recipients. The structured technique combining compositional approach with appropriate invariants and preconditions makes model checking of synthesised schemes unnecessary. Simulation results show that trusted paths through large endorsement trust networks can be generated at runtime using the hierarchical clustering technique devised. The proposed institutional trust mechanism promotes trust in e-commerce by endorsing both message originator and recipients. The notion of proof obligation and the techniques for discharging them allows derivation of schemes that make intermediaries accountable. The proposed framework combining endorsement trust with schemes making intermediaries accountable provides a way to alleviate distrust between previously unknown e-commerce entities.

Chapter 1: Introduction

The dramatic growth of e-commerce applications in the last decade can be attributed to the ready availability of information about products and services, increased trading options and reduced overheads. Ebay, for example, has made it possible for anyone to buy and sell many different categories of item without incurring excessive middleman costs. Businesses too have benefited greatly through more effective ways of conducting business-to-business transactions. Supply chains can now automatically take advantage of globally competitive prices using enabling technologies such as web service composition. Though emerging technologies allow services to be combined to meet the required functionality, at present end-to-end security and trust requirements are not adequately addressed [2]. Failing to address these real concerns can result in loss of revenue, loss of privacy, mistrust and increased cost [3]. Level of privacy is a major concern in e-commerce as any purchase related data may be distributed to other entities [4].

Protocols are simple programs that describe the order in which specific data elements must be exchanged between interacting parties to achieve specific goals. Security protocols use additional cryptographic elements and operations to meet various security goals such as data integrity and authentication. These security goals are met based on computational infeasibility of specific cryptographic operations without the necessary keys. Such goals are vital for e-commerce to ensure that interacting parties can be made accountable for their actions and to ensure confidential information is disclosed only to specific parties. E-commerce also requires protocols that provide end-to-end security assurances as data are often routed though one or more intermediaries.

A number of protocol suites have been devised to promote security and privacy in e-commerce applications in the last two decades [5-8]. For example, the SSL (TLS) suite provides privacy, authentication and integrity services between any two points [9], while application layer protocols provide end-to-end security for transactions involving electronic payments [6, 10-13]. The Internet Open Trading Protocol (IOTP) is a complex suite of protocols that defines transactions for purchase, refund, deposit, withdrawal and value exchange [8]. These protocols have specific strengths and have served well the type of configurations for which they were created. However, they are not designed to meet the goals and constraints of dynamically composed services where the type of resulting configuration is not known in advance [14]. For example, the SET purchase protocol requires messages between customer, merchant and payment gateway to be passed with specific encryption in a predefined order assuming standard goals for interacting entities. However, a specific merchant may require all messages from unknown entities to be routed through trusted intermediaries as a risk mitigation strategy, while the customer may want a greater level of privacy depending on the type of transaction. Collaboration between such entities is possible only if custom designed protocols can be synthesised based on functional and non-functional requirements [15-17].

Manually designing even simple security protocols is known to be error prone [18]. Hence, development and adoption of new security protocols is notoriously slow, often lasting many months, or even years [19]. It involves hand-crafting successive versions and verifying them for security assurances. Though model checkers can help in verification, they cannot automatically rectify the errors detected. Furthermore, existing model checkers suffer from combinatorial explosion as the number of paths an adversary can exploit increases exponentially with the number of steps in the protocol. These problems are exacerbated in e-commerce, where requirements are becoming more complex, often in response to government legislation and customer demand for increased accountability and privacy. Existing verification techniques can only be used in very simple protocols unless many simplifying assumptions are made to reduce their complexity [6, 11, 12]. Analysis of past protocol errors has revealed many protocol attacks could have been thwarted by using some common constructs [20, 21]. Using such constructs during design can reduce the number of revisions required and, consequently, the time period for adoption of new protocols. Therefore, research interest in security protocols has shifted from purely verification techniques to include creation techniques that prevent known attacks, and techniques that allow safe composition of existing protocols [22-26].

The full potential of e-commerce for collaboration is not fully exploited mainly due to the lack of trust promoting mechanisms [27]. Traditional commerce has used endorsements and guarantees to establish trust relationships with unknown parties. Intermediaries are often hierarchically organised where lower level endorsements are necessary before obtaining higher level ones. E-commerce too can benefit from such intermediaries that can provide the necessary endorsements and guarantees [28, 29]. Such intermediary selection is possible only if they are organised according to the types of endorsements they provide; a credit agency may endorse an entity for its creditworthiness but not character. Furthermore, trust in traditional commerce is a non-binary relation where the level of trust built up often determines the type of transaction that can be undertaken. E-commerce too, can profit from multilevel trust, if trust policies can be devised to regulate the extent of trust relationships based on past conduct.

If intermediaries are to be made accountable for their actions, security protocols should incorporate explicit evidences [30]. For example, an intermediary may prove that it endorsed data only after it was endorsed by a previous intermediary along the data path, or that it sent data authenticated only after receiving it authenticated. An endorsement intermediary directly endorsing data must be able to prove that its endorsement follows the predefined rules for such a data category in order to discharge its liability. Any intermediary failing to discharge its proof obligation may be penalised by having their trust relationships lowered or severed.

The difficulty in creating security protocols often stems from the difficulty of specifying the requirements precisely [31]. Even the flawed Needham-Schroeder protocol was a result of

misinterpretation of initial assumptions by protocol designers [32]. In defence of their protocol, Burrows, Abadi and Needham reasoned that they had assumed that principals do not divulge their own keys. Failing to clearly document design phase assumptions has made verification of e-commerce protocols difficult [6, 32]. Also, many of the security properties such as authentication and non-repudiation lack a standard definition [33]. Thus, before any attempt is made to automate the generation of security protocols, the precise meaning of security properties must be defined.

Security properties such as receiver non-repudiation, confidentiality, integrity, fair exchange and data origin authentication are essential for e-commerce transactions. Attempts to graft security such properties at a later stage in the design have been largely unsatisfactory, primarily due to increased cryptographic overheads. Cryptographic overheads can be reduced if the cost of security is used as a criterion in the service selection stage itself. Moreover, lack of fine-grained security schemes also lead to increased security overheads as a higher level of security assurance is provided than strictly necessary. If the e-commerce user base is to be extended to mobile and wireless devices, security concerns must be balanced with performance issues. Wireless applications, for example, must balance the risk posed by open communication channels with modest processing capabilities of resource constrained devices [34]. This trade-off between security and cost can be made easier if the protocol costs (both line and processing costs) can be estimated as a function of the protocol security strength, based on the underlying cryptographic elements and schemes.

An extensive literature review failed to reveal any past protocol synthesis attempt combining trust, and security. Such an approach can make it easier to achieve the right trade-offs between security, trust and performance, though it requires a number of research challenges to be overcome. The next section presents a brief survey of existing e-commerce security protocols and the challenges facing them, which provide the main motivation for this research. The following four sections summarise the research questions, the outline of the solution, the benefits/rationale for the approach and the contributions respectively.

1.1 E-Commerce Security Protocols

Standard security protocols such as TLS provide assurances data authentication, data integrity and confidentiality [9]. Security protocols for e-commerce and distributed systems must also include other important properties such as entity authentication, non-repudiation and recency [9,48]. The entity authentication property (*EA*) allows an entity to authenticate itself to another. The data security properties include recency (*R*), data authentication (*A*) and data integrity (*DI*), secrecy (*S*) and receiver non-repudiation (*RNR*). The receiver non-repudiation property provides data originator non-repudiable evidence for data receipt, while all other properties provide assurance to the data recipient. The purpose and example for these properties are listed in Table 1.1.

Property	Purpose	Example
Data Authentication (A)	Provides the recipient of data the assurance that data originated from a specific trusted entity and that it is the intended recipient	An agent entity performing a transaction on behalf of a customer may want the assurance that data originated from a valid client and was directed to it (not to some other entity)
Entity Authentication (EA)	Provides assurance about the identity of an entity	Before a customer can forward a message to an authorised agent the customer may want the assurance that the agent is what it claims to be.
Data Integrity (DI)	Gives data recipient the assurance that data has not been tampered with during transit	An agent entity acting on behalf of a customer may want the assurance that data was not altered during transit
Recency (R)	Gives data recipient the assurance that data is not stale, i.e., data was sent within a specific timeframe	A stock-broker may want the assurance that the data containing buy/sell instructions is recent
Receiver Non-Repudiation (RNR)	Assures data originator that data receipt cannot be denied at a later time.	A company submitting a tender may want the assurance that the recipient cannot deny receiving the data
Secrecy (S)	Limits access to specific parts of data	A customer may send credit information to the bank indirectly through a merchant but without giving direct access

Table 1.1 Purpose of e-Commerce Security Properties

All properties other than secrecy are considered correspondence properties, and are enforced at message level. Standard protocols used for e-commerce can be classified into point-to-point and end-to-end protocols. Point-to-point protocols are applicable when data are exchanged between two parties only, while end-to-end protocols provide security assurances even when data is sent through one or more intermediaries.

End-to-End Security

End-to-end security is vital in e-commerce as messages are often sent through one or more intermediaries. In the case of SET protocols designed for e-commerce, end-to-end security properties are preserved by using dual signature schemes. Such schemes allow a customer to send data through a merchant without disclosing all the data elements. However, such protocols are restricted to specific configurations and security properties, making their applicability limited. Protocols synthesised in this thesis attempt to meet the security needs of all interacting entities regardless of intermediaries through which they are despatched, thus providing end-to-end security guarantees. It guarantees end-to-end security by assigning a security level for all data exchanges along the message which is a function of security requirements of all interacting entities.

DEFINITION 1.1 END-TO-END SECURITY

An entity A provides end-to-end security assurance $s \subseteq \{A,RNR,DI,S,EA,R\}$ to another entity B if the security assurance s is provided for each edge along the path from A to B.

Fine-Grained Security Properties

Aggregating security properties requires fine-grained security properties to be defined. For example, a message sender may require non-repudiation assurance while one message recipient may require the recency assurance while another requires the authentication assurance. Fine-grained security properties allow end-to-end security requirements to be met by allowing security requirements along the message path to be aggregated. The fine-grained properties proposed in this thesis are made up of a combination of basic and composite security properties providing every combination of common security properties. A protocol devised dynamically can provide such assurances only if provably correct security schemes exist that can enforce such security requirements at runtime. In this thesis such schemes are derived using a compositional approach.

DEFINITION 1.2 FINE-GRAINED SECURITY PROPERTIES

Fine-grained security properties are made up of all the elements in the powerset of all correspondence security properties of interest.

For example, if {A,RNR,DI,EA} represent the set of all correspondence security properties of interest then fine-grained security properties are made up of {}, {A},{RNR}, …. {A,RNR,DI,EA}.

Studying the merits and shortcomings of these standard security protocols which are presented next, provides a useful reference point for automatic generation of security protocols.

1.1.1 Standard E-Commerce Security Protocols

The most widely used security protocols among those developed over the last two decades include Secure Socket Layer (SSL), Secure Electronic Transaction (SET), Internet Protocol Security (IPSEC), and Internet Open Trading Protocol (IOTP) among others. Two typical protocols have been chosen to illustrate the role of security protocols. SSL, a widely used protocol that provides point-to-point security, is presented first, followed by SET, a well-known application layer protocol designed to provide end-to-end security. These two protocols are typical of transport layer and application layer security protocols, providing point-to-point and end-to-end security. Though these protocols were designed independently, many implementations combine these protocols [9].

1.1.1.1 The SSL Protocol (now known as TLS)

The SSL/TLS [9] protocol was originally developed by Netscape to enable communication between web server and web browser. SSL allows peer processes on network devices to exchange messages securely via sockets. With respect to the OSI reference model, SSL lies between the transport and application layers as it runs under application layer protocols such as Hypertext Transfer Protocol (HTTP) and above the transport control protocol (TCP), as shown in Figure 1.1. An SSL session is an association between two entities that have a common set of cryptographic attributes and parameters. An SSL session is started whenever a client connects to a server, and it can remain active for up to 24 hours. In the first phase of SSL, the identification of parties, negotiation of attributes (such as the

hashing algorithm) and exchange of keys take place. In the second phase, data are exchanged using negotiated attributes and parameters.

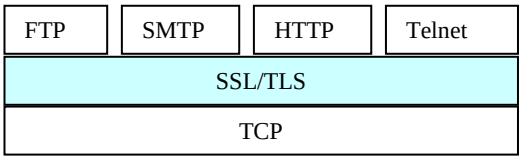


Figure 1.1 Hierarchy of Protocol Layers

Even though commonly used, SSL has a number of shortcomings. It is vulnerable to session hijacking, where the bridge between encrypted and non-encrypted parts in HTTPS is attacked. It lacks receiver non-repudiation service, which requires evidence of receipt that can be presented to third parties. It also lacks the fair exchange property, which ensures that no party gains unfair advantage when an e-commerce transaction is aborted half-way.

1.1.1.2 The SET Protocol

The SET protocol is an application layer protocol specification jointly designed by Visa and MasterCard to secure bankcard transactions over open networks such as the internet. It is a large protocol with many optional features and with documentation amounting to over 1000 pages. SET introduced two new entities into the architecture: a certifying authority and a payment gateway as shown in Figure 1.2. The certifying authority validates the actors and the payment gateway acts as an intermediary between the internet and the banking network that links acquirer and issuing institutions.

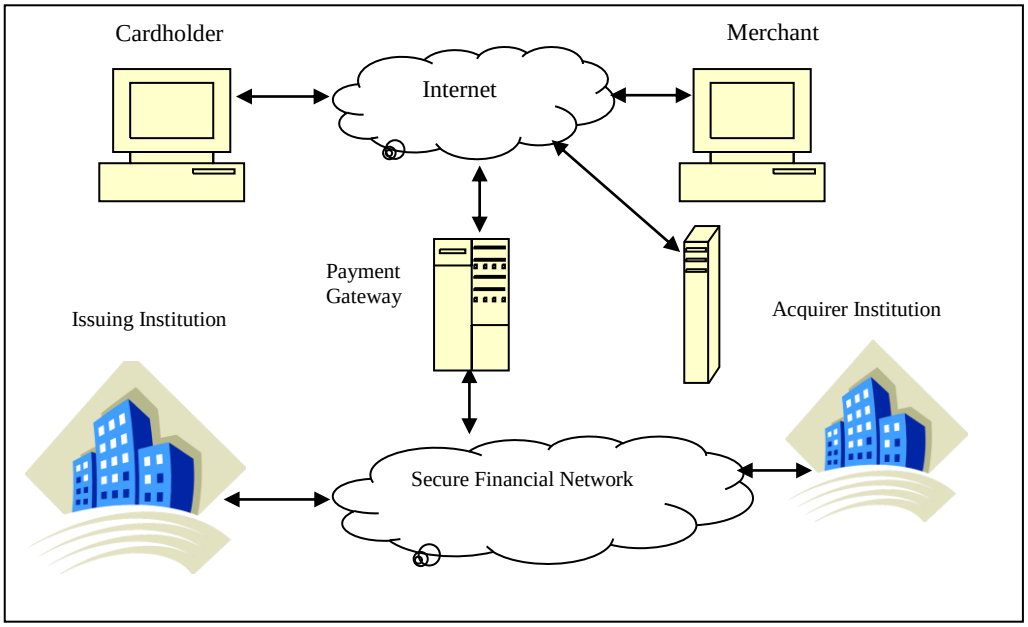


Figure 1.2 Entities in SET Protocol

The SET protocol consists of sub-protocols for the following transactions.

- Registration of cardholder
- Delivery of certificates
- Purchase transactions
- Payment authorisation
- Payment capture to initiate clearance on behalf of merchants

End-to-end security requirements are handled in SET by allowing integrity verification of data elements without revealing the contents. Semantic constraints are met by linking together specific data elements and by passing them in a specific order. For example, SET allows a customer to send purchase-order details and payment instructions together to the merchant with the implicit understanding that the payment instruction is to be forwarded to the payment gateway only if the merchant can meet the purchase-order requirements. Though the merchant can verify the integrity of the payment instructions received, it cannot access the details contained therein. This feature is enforced in SET using a dual signature, which requires signing hashes of data elements by multiple parties. The SET protocol, though effective in meeting end-to-end properties, is limited to bank transactions. If security protocols are to be widely applicable for e-commerce, interacting parties must be allowed to specify end-to-end requirements.

1.1.2 Past Research and the need for a Holistic Approach to Synthesis

E-commerce and web services must synthesise protocols dynamically if entities are to interact flexibly. Traditionally, e-commerce protocols such as SET were represented as a sequence of messages without explicitly capturing the underlying semantics of messages and their dependencies. Use of such protocols however, does not allow emerging opportunities through changing market conditions to be exploited. In a dynamic setting, the extent and types of non-functional properties must reflect evolving trust relationships. For example, the level of security assurances needed between entities may vary depending on their past relationships. Though some recent work has captured the semantics of messages through commitments and AI planning techniques, non-functional aspects have not been addressed [1, 94, 95]. Others have focussed exclusively on non-functional aspects such as end-to-end trust, security and response time without considering the underlying semantics of messages [2, 35, 36]. Others modelling non-functional attributes have taken a piecemeal approach, considering only one issue at a time, such as trust [17, 37-42] or security [2, 15-17, 43]. If protocols are to be generated dynamically they must be made to work within resource and bandwidth constraints imposed by underlying devices and mediums. None of the past synthesis approach surveyed model such trade-offs. These and other challenges are presented in greater details in Section 3.4.

1.1.3 Difficulties Creating Security Protocols for E-Commerce

The difficulties with designing and analysing security protocols stem from a number of considerations:

- Many of the properties that security protocols are designed to provide are extremely subtle. Some security properties such as authentication allow multiple interpretations. Basic properties, defined in Chapter 3, are combined to create more complex properties with precise meanings.
- A structured approach to composition is difficult as it must rule out any adverse interactions. Two protocols providing different security properties cannot always be safely combined to provide multiple properties. These problems are addressed in Chapters 3 and 5.

- Security protocols often operate in a hostile environment where adversaries mount attacks using their capabilities and the messages they intercept. Some common attacks, and techniques devised to counter these attacks, are presented in Chapter 3.
- If e-commerce entities are to be made accountable, security protocols must incorporate explicit evidence for third parties. Techniques to enforce accountability are presented in Chapter 5.
- Many e-commerce security problems are the result of underlying trust model which only allows an entity to completely trust or completely distrust another. Such a binary trust model is inadequate to represent the trust relationships in real life where different levels of trust exists [44]. A new non-binary trust model is presented in Chapter 4.

These factors highlight the need for a structured approach using a common framework that models security components, underlying trust relationships, workflow restrictions, semantic constraints and data dependencies.

1.2 Challenges Facing Synthesis of E-Commerce Protocols Incorporating Security and Trust

Section 1.1.2 identified some of the shortcomings in past e-commerce protocol synthesis attempts, and in particular, the need for a holistic approach taking into consideration both functional and non-functional requirements. This section summarises some of the challenges that must be overcome before such a holistic approach can be devised incorporating business requirements as well as security and trust considerations.

Aggregating Security Requirements

If protocols are to be created dynamically the security properties for each message must be set aggregating the security goals specified by the data originator and recipients. Currently basic and composite security properties lack a standard definition [33, 47, 48]. Basic security properties usually provide assurances to either the data originator or the recipient. Composite properties formed can be easily misinterpreted unless the order in which basic properties must be met is explicitly defined. Lack of well-defined security properties was a major cause of past security protocol flaws [31].

Composing Cryptographic Schemes

Unlike other areas of computer science, security schemes are less amenable to a compositional approach; two security schemes meeting distinct goals are known to interfere with the goals of one another when directly composed [26]. Security schemes for basic properties, however, can be composed to arrive at valid schemes for composite properties only when they are non-interfering. To allow composite schemes to be created for every security level, schemes should be made non-interfering, and where this is not possible, the conditions under which they are composed should be made explicit.

Security and Performance

Security requirements and performance constraints vary with types of network, device characteristics and transaction values. Performance bottlenecks are introduced by additional cryptographic elements that increase the bandwidth and the processing cost in terms of time and CPU cycles. Finding the right trade-offs requires devising schemes with cryptographic elements with varying security-strength to cost ratios. Comparing protocol costs requires a cost model parameterised in terms of underlying elements [49, 50].

Strengthening Protocols Against Common Attacks

Security protocols are vulnerable to a number of subtle attacks. Wireless and mobile devices open up many new avenues that can be exploited by adversaries [51, 52]. Preventing or reducing such attacks requires strengthening security schemes with additional counter measures, making the cost of attacks very high.

Modelling Trust in E-commerce

While cryptographic schemes can verify an identity or prove the origin of a resource they cannot prevent a dishonest-merchant supplying faulty goods. Cryptographic schemes therefore must be combined with intermediary mechanisms to promote trust between unknown entities [42, 53, 54]. Trust is central to traditional commerce. Many trust promoting mechanisms have evolved over the years. Trust in the physical world is considered to be directed, non-binary, category specific and dynamic. It is directed because *A* may trust *B* to pick good stocks, although the reverse may not be true. It is dynamic because *A* may stop trusting *B* after an adverse incident. It is category specific in the same way that one may trust a mechanic to repair a car but not a plumber. Trust is non-binary as a company may extend a credit limit up to $100,000 for one customer and only $10,000 to another. In e-commerce where there are no direct contacts, trust capital which reflects past experiences plays an even greater role. However, none of the protocol synthesis techniques have taken a holistic approach combining security aspects with dynamic trust relationships.

Need for End-to-End Security Schemes and Accountability

Before e-commerce can use one or more endorsement intermediaries, end-to-end schemes that make all intermediaries accountable for their actions must be devised. Many of the e-commerce security protocols lack an accountability mechanism that provides the means to associate an action with a specific entity [30]. Enforcing accountability requires incorporating sufficient cryptographic evidence to be presented to a third party adjudicator to prove misconduct, similar to audit trails in traditional commerce. Lack of such end-to-end schemes poses major challenges in e-commerce and web services.

Protocols for E-Commerce

Design of security protocols for e-commerce has proven difficult due to the interaction between security mechanisms and features necessary for e-commerce. For example, fair exchange feature requires no e-commerce party gain undue advantage at any stage of the protocol.

Synthesis problems combining security, trust and performance are difficult to solve as they require a multi-objective search criteria. However, such problems can be solved using a layered approach if they can be reformulated into disjoint sub-problems [55]. Though such techniques cannot guarantee optimality, they can help find feasible solutions.

1.3 Research Questions

Protocol synthesis problems have posed many challenges which researchers from different disciplines including distributed systems [56-58], artificial intelligence [57, 59], security protocols [23, 25, 60, 61] and e-commerce and web services [15, 35, 61-63] have attempted to address. This thesis addresses research questions related to synthesis of end-to-end security schemes through trusted endorsement intermediaries. Section 1.3.1 lists the main research questions pursued in this thesis.

1.3.1 Main Research Questions Pursued in this Thesis

Fine-grained Security Properties and Provably Correct Schemes (Chapter 3)

- E-commerce protocols (such as SET Purchase protocol) are designed to provide end-to-end security guarantees for all interacting entities even when data is sent through third parties. Before such protocols can be synthesised dynamically the following research question must be answered: How can end-to-end security requirements be made a function of security requirements of all interacting parties?
- A modular approach to security schemes has proven to be difficult as even valid schemes are shown to interfere when directly combined, leading to the research question: How can a structured approach be devised to create provably correct schemes for fine-grained security properties?
- E-commerce and web services require proven schemes to despatch messages to multiple recipients in a specific order, leading to the research question: How can two-party schemes be safely extended to any number of recipients?
- If protocols are to be synthesised to work within device constraints, security performance trade-offs are necessary, leading to the research question: How can the cryptographic elements with highest security strength that work within cost constraints be selected?

A New Institutional Framework for Promoting Trust between Unknown Entities (Chapter 4)

- Traditional commerce flourished only after institutional trust promoting mechanisms such as endorsements were introduced. This leads to the research question: How can an institutional trust framework for e-commerce be designed which allows vital data to be endorsed by authorised category specific intermediaries?
- If trusted paths are to be generated at runtime, trust relationships must be maintained centrally. In traditional commerce, existing trust relationships continue to evolve through direct experience while new trust relationships are established through recommendations. This leads to the research question: How can a centralized trust network allow establishment and evolution of trust relationships?

- Trusted paths for e-commerce may require data to be passed through large networks involving many hierarchical domains. This leads to the research question: How can the algorithms and data structures used for generating trusted path be made scalable?

End-to-End Security Schemes (Chapter 5)

- The validity of a message sent through multiple endorsement intermediaries depends on integrity of data and endorsements along the path. This leads to the research question: How can end-to-end schemes be devised that provide both data and endorsement security?
- When messages are sent through multiple endorsement intermediaries, misconduct by any entity or intermediary may cause a transaction failure. This leads to the research question: How can end-to-end schemes be devised that make all endorsement intermediaries and trading entities accountable for their actions?
- Protocols must be synthesised at run time if protocol paths through intermediaries are to reflect existing trust relationships. This raises the research question: How can newly synthesised protocols be distributed and enforced by all entities and intermediaries along the protocol path?

1.4 The Approach Proposed in this Thesis

Many current security challenges cannot be solved unless the underlying model combines the security architecture with trust based mechanisms. For example, mobility features make it difficult to verify the identity and the origin of a mobile agent unless a trust-enhanced security architecture is devised where such agents can only be received through specific other entities [170]. Similarly, end-to-end auditing is critical in service oriented architectures where services use a number of other heterogeneous services. Securing such services requires combining existing trust relationships with end-to-end accountability schemes [169]. E-health collaborations too, depend on combining trust establishment through intermediaries with security features, before common resources can be shared [168]. The underlying security features can vary from domain to domain but common features include data integrity, authentication, privacy and secrecy.

To address such problems this thesis studies the feasibility of creating a structured approach for creating end-to-end accountability schemes through trusted intermediaries with the required security properties at runtime using:

- a security framework that employs provably secure two-party schemes that provide the exact security requirements specified.
- a framework that establishes trust by allowing key data to be sent through category specific endorsement intermediaries with the required trust relationships.
- a framework that allows end-to-end security schemes to be derived and enforced that makes data originator, recipients and intermediaries accountable.

The synthesis problem is decomposed and solved using 3 distinct layers, as shown in Table 1.2.

No.	Layer Name	Layer Functionality
1	Fine-grained Security Layer	Defines fine-grained security properties and devises two-party security schemes that are provably correct.
2	Trust Establishment Layer	Creates trusted paths through category specific endorsement intermediaries.
3	End-to-End Security Layer	Derives provably secure end-to-end security schemes that make all intermediaries along the trusted path accountable.

Table 1.2 Layers in the Framework

The Fine-grained Security Layer (Layer 1 in Table 1.2) focuses on developing provably secure schemes for enforcing basic and composite security properties. The main steps in the proposed solution are outlined below.

- Basic security properties of interest are identified and classified into correspondence properties (for which there is a one-to-one mapping between send and receive events) and non-correspondence properties such as secrecy. Correspondence properties are enforced at message level while secrecy is enforced at data element level.
- Basic security properties are made to include features necessary for e-commerce such as fair exchange.
- The correspondence properties are combined to form fine-grained security levels. Any emergent behaviour required for composite properties is made explicit.
- The invariants and assumptions necessary for enforcing basic properties are made explicit.
- All security schemes are strengthened to withstand common attacks such as replay and type-flaw attacks.
- Security schemes for multiple properties are formed by combining the schemes for basic properties when their assumptions and invariants are not in conflict. Otherwise, non-interfering schemes are created before combining them.
- Basic schemes are proved using an approach similar to that used in secure protocol composition logic (SPCL) presented in Section 2.3.2.2. The validity of composite schemes follows directly from the compositional logic used to combine non-interfering basic schemes.
- Security schemes for multiple recipients are created by interleaving two-party schemes.
- Computational overheads for correspondence properties are estimated at message level while secrecy is estimated at data element level.

The Trust Establishment Layer (Layer 2 in Table 1.2) establishes trust between unknown intermediaries by exchanging key messages through category specific endorsement intermediaries. The main steps in the proposed solution are outlined below.

- Intermediaries are classified according to the data categories they are authorised to endorse. Intermediaries may perform direct endorsements based on knowledge of data elements or indirect endorsements based on trust placed in other intermediaries.
- A heuristic algorithm devised with varying weights for trust, performance and cost parameters helps create data paths with the required trade-offs.
- A non-binary trust model allows for growth/decay in trust relationships based on group and individual policies. Group policies reflect past positive experience, while individual policies reflect the trust disposition of entities for the specific domain and category.
- Large trust networks are decomposed into smaller domains to allow for efficient trusted path retrieval. Retrieval times can be further reduced by creating trusted paths offline.

The End-to-End Security Layer (Layer 3 in Table 1.2) creates the necessary end-to-end security schemes that allow intermediaries to be held accountable for their actions.

- End-to-end schemes are derived by combining the cryptographic evidences needed for intermediaries to transfer, delegate or discharge proof obligations. Intermediaries failing to produce the necessary evidences are considered to have breached trust assumptions.
- All messages passed through category specific intermediaries contain a variable part, in addition to the fixed part containing the data from originator. The data in the variable part can be altered by each subsequent intermediary reflecting its own endorsement.
- All schemes include a server signed digest made up of valid trusted paths, data category and a hash of original data. These server signed elements allow the path through which data are received to be validated by intermediaries and recipients. Both source and destination proof obligations are discharged only after validating the intermediary responses with server signed digests.
- The minimum security level along any edge in the data path is determined based on the end-to-end security requirements of data originator and recipients using that path. The server annotates this security levels along each edge in the trusted path generated.

1.5 The Benefits/Rationale of the Proposed Framework

The main benefits of the overall system and each individual layer are outlined briefly in this section. The overall benefits result from the holistic approach used in this framework modelling trust and security issues together. Benefits from each individual layer result from techniques devised to specify and enforce trust and security requirements.

Overall Framework

- This framework makes end-to-end security solutions possible by modelling the cyclic relationship that exists between trust and end-to-end guarantees. End-to-end guarantees require selecting intermediaries with good trust relationships (which reflect their past conducts), while trust relationships are adjusted based on the type of trust breaches detected by the end-to-end accountability schemes.
- Endorsements and guarantees are an essential part of institutional trust in traditional commerce. However, there has been no past attempt to dynamically select direct and indirect endorsements intermediaries based on category of e-commerce transactions. This framework allows key messages to be despatched through trusted category specific endorsement intermediaries, which provide a form of risk mitigation when interacting with unknown entities. The end-to-end schemes devised make all intermediaries accountable for their actions.
- The exact security level along the message path can be determined dynamically by aggregating fine-grained security requirements specified by interacting parties. The meaning of each fine-grained security property (formed by combining commonly used properties) is made precise by defining emergent behaviours, thus avoiding past protocol flaws through misinterpretation of security properties. The pre-created, provably secure cryptographic schemes for each fine-grained security property allow security protocols meeting requirements to be automatically synthesised.
- By modelling security and trust requirements at the data element level, this framework reduces the overall endorsement or security overheads. For example, when an entity sends a quote request that has no inherent value and involves no commitment, security assurances and intermediary endorsements need not be prescribed.

Benefits of the Fine-grained Security Layer

- Minimum security levels along the message path can be expressed as a function of the end-to-end security requirements of data originator and recipients, using the operations defined over fine-grained security properties. Enforcing the minimum security level helps to reduce the cost of security.
- The cost model allows computational costs and bandwidth of synthesised protocols to be used as the basis of selection when multiple protocols meet security and functional needs. Furthermore, by expressing the protocol cost in terms of underlying security strength of cryptographic operations, the model allows performance bottlenecks to be met (by lowering the security strength when necessary).
- The process of creating new protocols can be expedited by combining provably secure cryptographic schemes for fine-grained security properties with schemes that resist common attacks.

- The generic process devised for creating fine-grained security schemes may be extended to include other less common properties such as anonymity, subject to finding non-conflicting schemes.

Benefits of the Trust Establishment Layer

- The proposed architecture allows e-commerce entities to exchange vital messages through authorised category specific intermediaries. Such endorsements promote trust by providing a form of risk mitigation. Direct intermediaries endorse messages based on domain knowledge of entities, while indirect intermediaries endorse messages based on endorsements by trusted intermediaries.
- Trusted paths can be selected according to differing criteria that reflect the type of environment and the perception of entities. For example, in one domain low intermediary cost may be set as the prime factor by attaching to it a greater weight than transitive depth and trust relationships.
- A self-regulating trust network is made possible using policies that allow trust evolution and trust transfer. Trust evolution reflects past conduct, allowing trustworthy intermediaries to build up their trust capital over time. Trust transfer allows new trust relationships to be established, thus reducing the number of intermediaries required.
- Endorsement costs are lowered by sharing common endorsements for data sent to multiple recipients.

Benefits of the End-to-End Security Layer

- Intermediaries are made accountable using explicit cryptographic evidences in end-to-end schemes that can be presented to a third party. The ability to prove breach of trust provides the basis for revoking endorsement capability and lowering trust relationships, thus serving as a deterrent to misconduct.
- Intermediaries are allowed to verify that messages are despatched along trusted paths before endorsing them, using the server signed data path despatched with data.
- Fine-grained security provided for both data and intermediary endorsements along the data path help achieve better trade-offs between security and cost.

1.6 Contributions

Synthesising end-to-end security schemes through endorsement intermediaries required addressing a number of research problems. This section outlines the main contributions.

A New Mechanism for Quantifying and Reasoning About Security Requirements

Fine-grained security properties define hierarchical security levels using a lattice model where each security level represents a composite security property. These composite properties are formed by combining basic security properties needed by data originators and recipients. The meanings of composite properties are made precise by standardising the order in which individual properties must be enforced. The operators defined over them allow overall security requirements along the path to be expressed as a function of security requirements of interacting parties. By defining all security properties in an unambiguous way many of the past security flaws caused by multiple interpretations of common security properties can be avoided. The hierarchical structure makes it possible to specify exact security requirements, thus avoiding overheads caused by over-prescription of security properties. Fine-grained security properties also make it possible to consider security as a core requirement when synthesising protocols for composed services. Past synthesis techniques modelled functional requirements or underlying semantics [73,74,94,95] in isolation without considering the security needs. The proposed model allows functional requirements and semantic constraints to be combined with security needs using the fine-grained security properties and schemes devised.

Provably Secure Schemes Providing Fine-grained Security Properties

A new technique was devised that allows the dynamic creation of security protocols by combining the provably secure schemes providing fine-grained security properties. Provably correct schemes enforcing these properties were derived by combining challenge-response mechanism with compositional logic. Deriving schemes for composite properties from common base schemes makes it possible to sequentially compose them. Though a number of previous attempts have been made to synthesise protocols using search based mechanisms [23, 24, 73] or composition of protocols [61, 75], which are discussed in detail in Section 2.3. However, no past attempt was made to combine these techniques to create fine-grained security schemes that provide the exact security level required. It was also shown that these schemes can be combined with techniques that resist common forms of attack such as replay and type-flaw attacks.

Extending Two-Party Schemes using Interleaving Technique

A novel interleaving technique devised allows any two-party scheme to be extended to multiple parties. Using this technique all two-party schemes providing fine-grained properties were extended to multiple recipients. This interleaving technique was used to create schemes for multiple recipients by piggybacking data and cryptographic elements. The validity of extended schemes was proved using Strand Spaces. The interleaved schemes provide a number of benefits. First, direct evidence from

originator and recipients are included even though data is sent through third parties. Second, the number of protocol steps required at data source is significantly reduced when data must be despatched to multiple recipients with various security properties.

Security Performance Trade-Offs

The proposed model for protocol synthesis facilitates security performance trade-offs by allowing protocol costs to be expressed in terms of underlying security strength. Security strength gives a measure of processing effort required by an adversary to break the underlying mechanism. Cryptographic algorithms, elements and key lengths can be grouped together to provide different levels of security strength. Such an approach makes it possible to lower security strength to meet the underlying performance and cost constraints. Past attempts to synthesise security protocols did not consider such trade-offs as security was modelled only in terms of symbolic logic without any consideration for underlying cryptographic elements. However, the pervasive use of security protocols across different networks and devices makes such trade-offs inevitable.

Framework for Institutional Trust

A new institutional trust framework proposed can help overcome the perception of risk in e-commerce. The endorsement intermediary network consists of nodes representing trading entities and authorised intermediaries, while edges represent the extent of trust relationships that exist between them. All intermediaries are made accountable for their endorsements. The path endorsement trust proposed gives a measure of indemnity based on the number of endorsement intermediaries along the path and the extent of the trust relationships that exist between them. By combining category specific trust and endorsements with security mechanisms for detecting trust breaches, the proposed framework provides a much stronger basis for trust propagation than the transitive trust models proposed in the past.

Finding Trusted Paths through Large Trust Networks

Hierarchical domains and domain trust trees devised allow trusted paths through large networks to be found at runtime. Search space and time are reduced logarithmically by subdividing large networks into domains of limited size. Elapsed times are further reduced by using pre-created domain trust trees with different criteria. Simulation results show that for each domain with medium trust coupling the average search time grows linearly. Estimates based on these results suggest that trusted paths can be retrieved within one second, even for networks of up to 800,000 nodes. If endorsement costs are to be reduced depth-limited paths must be found efficiently. A search criteria devised combining path trust with transitive-depth reduces the number of intermediaries (and endorsement costs) significantly. Simulation results also show such algorithms result in paths to more entities than those algorithms using path trust alone.

Policies for Self-regulating Centralised Trust Networks

The centralized trust network is proposed models trust as a derived value based on past conduct, trust disposition and trusting beliefs. Trust transfer and trust evolution policies proposed makes the centralized trust network self-regulating. Trust evolution policies either raise or lower trust relationships based on transaction outcomes. The extent of these changes reflects trusting beliefs in underlying domains for specific categories. Trust transfer policies allow new trust relationships to be formed reflecting their own trust disposition. Simulation results show that trust transfer threshold should be varied over time if entities are to build and preserve their trust capital. These results suggest that trust in traditional commerce can be modelled more closely by combining trust evolution policies with trust transfer policies.

Deriving End-to-End Accountability Schemes

The notion of proof obligation devised facilitates reasoning about accountability for various security properties. End-to-end accountability schemes for security properties providing assurances to recipients (A,DI,TB) can be derived by transferring proof obligations through intermediaries until they can be discharged using direct evidence from the data originator. Similarly end-to-end accountability schemes for security properties providing assurances to data source such as receiver non-repudiation (RNR) can be derived by delegating proof obligations through intermediaries until direct evidence is obtained from all final recipients. The past research with accountability discussed in detail in Section 2.5, was mainly restricted to verifying whether existing schemes makes protocol participants accountable.

Enforcing End-to-End Security

An SSES scheme generates the protocol metadata for enforcing end-to-end security at runtime. The server signed metadata consists of data hash, data category, trusted path and the security-levels required along the path. Each intermediary and recipient along the trusted path is required to enforce the security levels specified using the proven two party schemes in their possession. Path security facilitates hierarchical endorsements common in traditional commerce where endorsements must be carried out in a predefined order.

1.7 Thesis Structure

The rest of the thesis consists of background, contribution and conclusion chapters as follows:

Chapter 2 presents background information related to this thesis.

Chapters 3, 4 and 5 (Contribution Chapters)

The main contribution of this thesis titled "**Synthesising End-to-End Security Schemes through Endorsement Intermediaries**" is a framework for promoting e-commerce between unknown entities by allowing important messages to be sent through endorsement intermediaries with user specified security properties. The first contribution chapter focuses on creating the security schemes dynamically based on requirements. The second contribution chapter focuses on selecting trusted paths through endorsement intermediaries based on message contents. The third contribution chapter focuses on making entities and intermediaries accountable, using explicit cryptographic evidence**.**

- **A new Framework for Generating Security Protocols using Proven Schemes Devised to Enforce Fine-Grained Security Properties is presented in Chapter 3**. This contribution chapter first defines basic and composite security properties of interest for e-commerce. Schemes for these properties are devised using challenge-response mechanisms and compositional techniques. Next, schemes for single recipient are extended to multiple recipients using interleaving techniques devised. These schemes are also strengthened to withstand common attacks based on analysis of past protocol attacks. Finally, standard protocols are created automatically using the schemes devised.
- **An Institutional Framework allowing Endorsement of Key Messages by Trusted Category Specific Intermediaries is presented in Chapter 4**. This contribution chapter proposes a framework to reduce the risks involved in trading with unknown e-commerce entities, by allowing key messages to be passed through trusted endorsement intermediaries. Trusted paths for messages are selected based on a number of different criteria, including the extent of trust relationship along the path, trust depth, and cost of intermediaries. Trust relationships are allowed to evolve on the basis of past conduct, both positive and negative. New trust relationships are established reflecting trust disposition of entities and intermediaries. Clustering and retrieval techniques are devised to generate trusted paths efficiently.
- **End-to-End Schemes for Enforcing Accountability are presented in Chapter 5**. This contribution chapter presents end-to-end schemes devised that make all intermediaries accountable. All data despatched are accompanied with server signed trusted paths and required security levels to ensure path security. All intermediaries are required to discharge their obligations by providing explicit cryptographic evidence that can be presented to a third party. Security levels specified along trusted paths are enforced using proven schemes that are sequentially composable.

Chapter 6 concludes the thesis by presenting the overall contributions and possible future extensions.

Chapter 2: Background and Mathematical Foundations

This chapter reviews the literature relating to security protocols and lays down the mathematical foundation required for the subsequent chapters. Much of the past research into security protocols has been devoted to various forms of verification technique. Two common theorem-proving approaches are described in detail. BAN logic allows the beliefs of agents to be modelled. Verifying protocols using Strand Spaces, a graph-based approach, requires proving a one-to-one correspondence between specific receive events and earlier send events initiated by a valid party. The model checking approach automatically tests whether the model of the system meets a given specification. This requires both the model of the system and the specification to be formulated in precise mathematical language. Past logics used for protocol composition are surveyed briefly before presenting Secure Protocol Composition Logic (SPCL), a temporal logic-based approach for modelling protocol composition. Common attacks on security protocols and techniques devised to prevent them are presented next, followed by accountability in e-commerce protocols. The last section presents the need for security performance trade-offs in protocols.

2.1 Security Protocols

The role of a security protocol is to ensure reliability and dependability of information exchanges through data authentication, data integrity, non-repudiation and secrecy [69]. The sequence of steps in a security protocol is designed to make it very difficult or impossible for an adversary to mount a successful attack. Designers must often revise protocols when they are used with different domains or devices, considering the level of threats posed and underlying constraints on resources. For example, many revised versions of the well-known SSL protocol were derived to suit wireless medium and mobile devices [64].

2.1.1 Main Elements of a Security Protocol

This section describes the main elements in a security protocol in terms of the roles played by the protocol entities and the purpose of common cryptographic components.

- **Principals:** the valid entities taking part in the protocol, including vendors, customers and trusted intermediaries. It is customary to use names such as Alice, Bob and Carol to refer to protocol participants playing various roles. It is possible for an entity to play multiple roles at the same time such as, playing the role of merchant in one protocol and customer in another.

- **Adversary:** any entity acting maliciously to subvert a protocol. A principal may act as an adversary if principals in a protocol do not trust one another. It is possible for a valid participant in a protocol to be an adversary in another run of a protocol. In such a case, information gained from one run may be used to mount an attack in the other.

- **Message contents (payload):** may include both business-related information such as product ID, customer ID, and credit card and cryptographic elements such as nonce, time stamp and key. The information may be sent as plaintext or in encrypted form.

- **Identifiers:** are used for principals, trusted intermediaries, the server and the adversary. Common labels assigned are principals *A, B;* the server *S;* trusted intermediaries *T1*, *T2;* and the adversary *Z.* The notation *Z(A)* is used when an adversary is impersonating a valid participant *A.*

- **Keys:** Cryptographic key systems are classified into symmetric and asymmetric key systems. In symmetric key systems, keys used for encryption and decryption are either identical or can be easily derived from one another. In asymmetric key systems, a pair of keys known as a public key and a private key is generated, with one issued for encryption and the other for decryption. Deriving one key from the other is possible but is computationally infeasible. Private keys should never be disclosed, while public keys can be made available to any other entity. A message encrypted with a key in one pair can only be decrypted with the associated key in the same pair. Therefore, a message sent encrypted with the private key of a principal can only be decrypted with the public key of that principal, proving the identity of message originator. Similarly a message sent encrypted by the public key of the intended recipient can only be decrypted by that entity (as private keys are never divulged), providing confidentiality.

- **Nonces:** Nonces are random numbers used for uniquely identifying a run or associating different protocol elements. The probability of two identical nonces being created independently using existing algorithms is so low that it can be assumed to be 0. Hence, they are used to provide assurance of recency or freshness.

- **Hashes:** A hash function *h* has the characteristic that makes computing hash of message *m, h(m),* easy but finding two distinct messages *m1,m2* such that *h(m1) = h(m2)* is computationally infeasible.

- **Protocol notation**: Each protocol step is represented in the form: $A \rightarrow B$: *message elements* Here *A* represents the sender of the message and *B* the recipient.

2.1.2 Security Protocol Assumptions

Validity of security protocols depends on the underlying assumptions. Proofs of security protocols are derived from these assumptions and other mathematical axioms. Common assumptions include:

- Perfect cryptography: encrypted items cannot be decoded within available resources and time by entities not in possession of the necessary keys.
- Binary trust relations: entities either fully trust one another or there is no trust relationship at all.

To illustrate the security challenges, the next section presents two standard security protocols. The subsequent section discusses the role of security protocols, including common security properties, assumptions under which they operate, and their vulnerabilities.

2.1.3 Example of a Security Protocol: Key Exchange Protocol

The Needham-Schroeder symmetric key protocol described below is designed to distribute a fresh key K_{PQ} between two principals P and Q. It is assumed that P and Q already share the symmetric keys K_{PS} and K_{QS} with the key server S. The protocol entities P and Q use the nonces n_P (generated in P) and n_Q (generated in Q) to confirm that the message responses are recent. In the first step below, P requests the server *S* to generate a symmetric key that can be shared with Q. In the second step, S responds with a message encrypted with the key it shares with P with two parts, both of which contain the fresh key K_{PQ}. The first part is intended for P, while the second part encrypted with the key shared by S and Q is intended to be forwarded to Q in the following step (step 3). In the fourth step Q challenges P with the nonce n_Q encrypted with the fresh key K_{PQ} it now shares with P, to which P responds in the following step using the same key K_{PQ}.

Message 1 $P \rightarrow S : P, Q, n_P$

Message 2 $S \rightarrow P : \{n_P, Q, K_{PQ}, \{K_{PQ}, P\}K_{QS}\}K_{PS}$

Message 3 $P \rightarrow Q : \{K_{PQ}, P\}K_{QS}$

Message 4 $Q \rightarrow P : \{n_Q\}K_{PQ}$

Message 5 $P \rightarrow Q : \{n_Q - 1\}K_{PQ}$

2.2 Protocol Verification

Protocol verification using formal approaches became popular after flaws were detected in fairly simple protocols. Formal approaches help by delineating system boundaries and by defining protocol goals and system behaviour. They also allow reasoning about intruder capabilities and actions. However, verification is undecidable unless specific assumptions are made about the number of possible interleaving runs, intruder capabilities and possible actions. [65]. An intruder may use some

initial knowledge to learn even more secrets as the protocol sessions proceed. The threat model proposed by Dolev-Yao in the early 1980s has become the standard model for formal analysis. It assumes perfect cryptography and a computationally unbounded adversary that has complete control over the network. The adversary is assumed to have abilities to eavesdrop, intercept and fake messages.

A formal verification technique has three main parts to it:

- a way to specify the requirements
- a framework for modelling the system
- a way to establish that the model enforces these requirements

Formal methods have been applied at different levels of abstraction using beliefs based logic (BAN), process algebra (CSP) and graph-based techniques (Strand Spaces) [66] [67]. Use of formal methods ensures that protocol design meets precisely defined protocol goals. Formal methods help to prove that adversary with specific capabilities and initial knowledge cannot undermine protocol goals.

2.2.1 Theorem Proving

The goal of theorem proving is to prove that the given protocol design precludes certain events from taking place. Such techniques are traditionally used for proving the correctness of programs. Unlike model checking, theorem proving places no bounds on the number of parallel executions (to avoid combinatorial explosion). Theorem proving helps deduce whether a formula F holds in all models (i.e., $\vDash F$) [165]. Proof inference techniques use a collection of inference rules of the form below to arrive at the required conclusion. The conclusion Con is guaranteed when all the premises $Pr1$, $Pr2$ are true or are derivable.

$$\frac{Pr1,\ Pr2}{Con}$$

Axioms, which are inference rules with no premises, are used in proof derivations. A number of automated theorem proving tools have evolved for verification using inductive proofs including ITP, Isabelle and PVS [68,69]. Although these methods are fairly general and applicable to a wide class of problems, they require a fair amount of user guidance.

2.2.1.1 BAN Logic

BAN logic, a form of modal logic, was developed by Burrows, Abadi and Needham [166]. This logic was presented in 1989, in one of the earliest papers that analysed security protocols formally. It received much attention as the language presented an elegant and intuitive way to model the beliefs of agents. Since then, it has received much criticism as protocols were analysed with the assumption, often used in those days that all valid participants are honest. It also has other limitations such as the

inability to represent the secrecy requirements of protocols. The semantics of BAN primitives are shown in the Table 2.1.

Primitive	**Semantics**
$P \mid\equiv X$	P believes X
$P \overset{k}{\longleftrightarrow} Q$	K is key shared by P and Q only
$\#(X)$	X is fresh (recently generated)
$P \triangleleft X$	P sees X (or P receives X)
$P \mid\sim X$	P has in the past sent a message containing X

Table 2.1 Semantics of BAN Primitives

In this approach, beliefs of each protocol participant are recomputed after each step, based on existing beliefs, message receipts and inference rules. For example, if P receives X encrypted with K_{PQ}, and it believes that Q is the only other entity in possession of that key K_{PQ}, then P can form the belief that Q once said X. This inference rule can be written in BAN logic as in:

$P \mid\equiv P \overset{K_{PQ}}{\longleftrightarrow} Q$ — P believes Q is the only other entity in possession of K_{PQ}

$P \triangleleft \{X\}_{KPQ}$ — P receives X encrypted with K_{PQ}

———————

$P \mid\equiv (Q \mid\sim X)$ — P believes Q once said X

Protocol analysis using BAN (whether a protocol meets its goals) requires the following major steps:

- conversion of initial beliefs into BAN statements
- conversion of protocol goals into BAN statements
- conversion of protocol steps into BAN statements
- application of inference rules to arrive at new beliefs
- comparison of final beliefs with protocol goals.

In the recent past a number of attempts have been made to synthesise protocols using BAN logic. These approaches are discussed in greater detail in Chapter 5.

2.2.1.2 Strand Spaces

Another model using the theorem proving approach is Strand Spaces, where protocols are proved by showing that specific invariants are never violated [69]. A strand represents the series of visible actions by a protocol entity such as sending and receiving messages. Strand Spaces consist of a set of strands representing the principals and adversaries involved in the protocol. The underlying graphical structures used for modelling events provide an elegant way to reason about the properties of security protocols.

Sending and receiving of messages is represented by positive (+ve) and negative (-ve) nodes respectively, as shown in Figure 2.1. If the nodes N_1 and N_2 have the relationship $N_1 \rightarrow N_2$, then these nodes have the form $N_1 = +a$ and $N_2 = -a$, indicating N_1 sends the message a to N_2.
Nodes that represent one event immediately preceding another on the same strand are connected by double arrows ($\Rightarrow$). If the nodes N_1 and N_2 have the relationship $N_1 \Rightarrow N_2$, they belong to the same strand with $index(N_2) = index(N_1) + 1$ (*index* refers to the order within the strand).

A bundle is a subset of a Strand Spaces representing all the strands in a protocol exchange. The edges of a bundle $\rightarrow$ and $\Rightarrow$ express the external and internal causal relationship of adjoining nodes respectively.

A message term is made up of text terms, keys and their combination. Text terms could include *principal-name* and *customer-ID*.

As an example, consider a simple protocol between two principals *A* and *B (*using the standard protocol notation*)* exchanging nonces using a shared key *k*.

$A \rightarrow B: \{N_a\}_k$

$B \rightarrow A: \{N_b\}_k$

$A \rightarrow B: N_b$

This protocol can be represented by a bundle consisting of two strands as in:

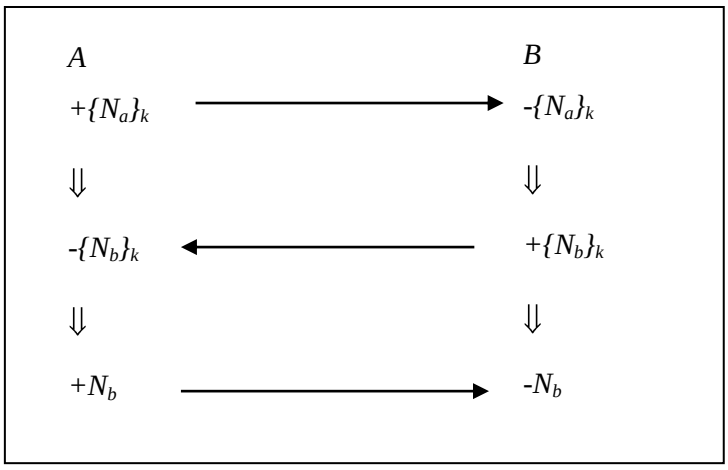


Figure 2.1 Strand Spaces Representation of Simple Protocol

Causal Precedence

Two nodes N_1 and N_2 in a bundle *C* are related by $N_1 \leq N_2$, if there is a sequence of zero or more edges of type $\rightarrow$ and $\Rightarrow$. The relation $\leq$ is a partial order, i.e. it is reflexive, anti-symmetric and transitive. Every non-empty subset in *C* (i.e., which has at least one edge of the form $\rightarrow$ and $\Rightarrow$) has at least one member.

Capabilities of Adversary Strands

An adversary can generate new messages from intercepted messages and its initial knowledge. These actions are modelled by a set of adversary strands, with one or more parameters representing data items (g,h,t) or keys (k, k^{-1}). Note, the adversary strands below contain both send (+) nodes and receive (-) nodes. The strand $T[g]$ for example, on receipt of term g sends it out to two other strands, while the strand $C[g,h]$ receives the terms g and h separately and sends out the appended term $g.h$. A protocol attack involves combining a number of these actions.

$M[t]$	Inset message: $<+t>$ where t belongs to the initial information of intruder
$G[g]$	Flushing: $<-g>$
$T[g]$	Tee: $<-g, +g, +g>$
$C[g,h]$	Concatenation: $<-g, -h, +g.h>$
$R[g,h]$	Separation into components: $<-g.h, +g, +h>$
$K[k]$	Key: $<+k>$ where $k \in K_p$
$E[k,h]$	Encryption: $<-k, -h, +\{h\}_k>$
$D[k,h]$	Decryption: $<-k^{-1}, -\{h\}_k, +h>$

Proving the Security Properties

General theorems about the power of an adversary are developed based on the capabilities of the adversary strands (defined above). These theorems, the invariants and nonces together are then used together to prove security properties such as authentication and secrecy. Authentication can be defined in terms of the agreement property [33] which states "A protocol guarantees a participant B (say as the responder) agreement for certain binding, if each time a principal B completes a run of the protocol as a responder using, supposedly with A, then there is a unique run of the protocol with principal A as initiator, supposedly with B". This requires proving that whenever a bundle C contains a responder strand using X made up of specific combination of nonces, the bundle C also contains a unique initiator strand containing X. The secrecy property is proved using the invariant that requires all secret data elements be encrypted by a key not known to the intruder, before being despatched.

2.3 Previous Work on Protocol Synthesis

Past research interest in security protocol synthesis was motivated by the long delays in the traditional process of crafting and verifying security protocols [15, 71, 72]. Approaches to protocol synthesis can be classified into search-based techniques and compositional approaches. An early attempt to automate protocol generation (APG) involved creating all possible two and three party protocols before discarding invalid ones using a fast model checker [73, 74]. Other approaches used meta-heuristic and backward search techniques to avoid combinatorial explosion [23, 24]. Another attempt breaks down protocol goals specified in BAN logic into sub-goals, which in turn determine the messages that must be passed between parties [60]. Protocol composition techniques have used Secure Protocol Composition Logic (SPCL), Strand Spaces and Pi-Calculus, a form of process calculus.

2.3.1 Search-Based Synthesis Techniques

2.3.1.1 Automatic Protocol Generation and Selection

Adrian Perrig and Dawn Song [73] devised a technique known as Automatic Protocol Generation (APG), which generates all possible protocols using the grammar below.

Message ::= *Atomic* | *Encrypted* | *Concatenated*
Atomic ::= *PrincipalName* | *Key*
Encrypted ::= (*Message, Key*)
Key ::= *PublicKey* | *PrivateKey* | *SymmetricKey*
Concatenated ::= *Message List*

Protocols exceeding a specified number of messages are discarded. All protocols are verified using a model checker named Athena [74]. Protocols failing to meet the goals are discarded. The main novelty of this approach is the use of an efficient model checker in selecting the protocol. However, its applicability is limited to two and three party protocols. The blind search technique cannot scale up to more complex protocols involving multiple entities and messages. The model checker Athena used for verification could only verify authentication properties. The protocols created by the grammar have no semantics attached to them, making it impossible to decide when they are applicable, without human involvement.

2.3.1.2 Backward Search from Protocol Goals

ASPB (Automatic Synthesis Protocol Builder) uses an automatic backwards search to synthesise security protocols from goals [23]. Given a number of distinct protocol goals, it creates separate sub-protocols before combining them. ASPB expresses the requirements specifications, the initial assumptions and the goals in BSW, a variation of BAN logic with channels similar to those used in SPI calculus. Heuristic rules for synthesis are also expressed in BSW, which allows goals to be expressed in terms of sub-goals, which may include message transfers through specific channels. A synthesis process involves using these heuristics recursively until goals can be reduced to assumptions or messages through specific channels. These messages are combined to form sub-protocols. The sub-protocols are then merged to form the overall protocol. In a similar attempt, high-level security goals [60] specified in BAN logic (refer to Section 2.2.1.1) are used in synthesis. It incorporates an inference engine that breaks down a set of goals into elementary goals, and a realisation function that maps an elementary goal into protocol actions. Its inputs are assumptions about the environment and initial beliefs of the entities. Unlike APG, this approach allows security properties' integrity, authentication, non-repudiation and confidentiality to be represented.

The backward search process together with the heuristics used for guidance help reduce the search space significantly. As a result, the time required for protocol generation is significantly lower than in

APG. Breaking down protocol goals using search techniques provides a top-down approach to protocol synthesis (from goals to sub-goals that can be enforced). The main drawback is that the method is not guaranteed to produce a protocol even when the requirements are valid, as the inference engine may not be able to break down all the goals into elementary goals. Furthermore, its applicability for e-commerce is limited as BSW and BAN logics deal with beliefs that cannot be produced as evidence. It is also an idealised protocol, since all entities are assumed to be honest. Furthermore, creating protocols at an abstract level without using explicit cryptographic elements, makes it difficult to model protocol costs.

2.3.1.3 Security Protocol Generation through Meta-heuristic Search

A meta-heuristic search allows a guided search of protocol design space (made up of all the combinations of protocols that can be created in a specified number of steps) for desired goals, starting with initial beliefs [24]. Current beliefs, goals and messages use BAN logic operators such as *sees*, *believes*, *has-jurisdiction* and *fresh*. Communication based on the current beliefs of the sender cause new beliefs to be formed in the recipient using inference rules defined in BAN logic. The search strategy selects sender, recipient and message elements randomly, from existing participants and beliefs respectively. The fitness function for guiding the search is a function of the number of protocol goals achieved and the weight attached to goals. Varying the weights with time influences the number and type of protocols created. In the early credit scheme the weights are constantly reduced, giving greater rewards for earlier messages satisfying the goals, while with uniform credit all weights remain unchanged.

This approach often results in creating protocols that meet all the goals. However, the main weakness of this approach is that the success rate and the time for synthesis can vary significantly with different weighting strategies. The ideal weighting strategy for a given domain can only be determined by experimentation. It is also possible that none of the existing weighting strategies may be appropriate for a specific setting. These drawbacks make such an approach unsuitable for e-commerce, where interaction protocols must be created dynamically.

2.3.2 Compositional Approach to Synthesis

2.3.2.1 Strand Spaces Based Composition

In a more recent work, composition of protocols expressed as Strand Spaces has been attempted [75]. The mechanisms known as authentication tests [47] use a number of challenge-response schemes associated with security goals to design sub-protocols manually. These schemes consist of a series of messages, with data in either encrypted or in plaintext form. Public key encryption is used to assert data origin and to restrict access, while nonces are used to indicate recency or to show association with earlier messages. Hash values are used in acknowledgements to reduce the bandwidth required.

The first test, known as an incoming test, allows an inference to be made about the existence of an entity with key *K*, whenever a message sent in the open is later received back encrypted with a key *K*, (where $K \in S$, S is the set of safe keys). Similarly, an outgoing test allows an inference to be made when a message sent encrypted is later received in the open (unencrypted). Using these tests, two-party protocols were devised to meet authentication and non-repudiation goals. Sub-protocols created using authentication tests were combined to create new protocols. Use of distinct cryptographic elements in tests ensures protocol goals are not breached when combined [76].

Evaluation

The graph based structure used in Strand Spaces allows reasoning about partial ordering of messages and origin of messages. Thus, it is well suited for reasoning about capabilities of adversary strands. Authentications tests, a form of challenge-response mechanism, uses one-to-one mapping between entities based on private and public key pairs. Combining these authentication tests provide a strategy for creating new schemes. The Strand Spaces approach however, is not well suited when branching behaviour is required, as the principle of unique origin no longer holds. Furthermore, Strand Spaces do not explicitly represent data known to a strand. Thus, end-to-end schemes where messages are passed through one or more intermediaries are difficult to design using authentication tests alone.

2.3.2.2 Secure Protocol Composition Logic

Secure Protocol Composition Logic (SPCL) [61] uses Cord Spaces, a process calculus based on the process algebra PI Calculus [70]. The logic provides the means to reason about protocol traces using axioms, assertions and inference rules for each of the main protocol actions. SPCL uses assertions similar to those in Floyd-Hoare logic, which allows proof for protocol correctness to follow from the way it is composed [77]. Hoare's logic uses expressions of the form: $\{P\}\ C\ \{Q\}$ where P and Q are predicates and C is a set of instructions. It asserts that whenever C is run in a state where P is true, Q will hold when C terminates. By limiting its actions to those that do not violate specified invariants, it allows new security guarantees to be derived. SPCL uses two basic forms of composition technique to create more complex protocols out of simple ones.

The first form, additive combination allows two protocols to be sequentially composed. Given the predicates ϕ and ψ, the action P and the entity A, $\phi[P]_A\psi$ means that if ϕ is true before A performs actions P, then ψ is true afterwards. For example, if P is the action *"A* receiving a signed message from *B"*, and ϕ the precondition that *"A* is in possession of *B*'s public key*"*, then the post condition ψ asserts *"B* sent the signed message to *A"*. These "before" and "after" assertions allow individual protocol actions to be combined to derive assertions for a sequence of steps. For example, $\phi[P1;P2]_A\theta$ can be derived if $\phi[P1]_A\psi$ and $\psi[P2]_A\theta$ are both true.

The second form of combination requires identifying invariants that guarantee before and after assertions. If such an invariant Γ can be found for protocol *P1* with pre and post conditions ϕ and ψ, it can be expressed as $\Gamma \vdash \phi[P1]_A\psi$. This says that $\phi[P1]_A\psi$ holds in any run satisfying the invariant Γ.

For example, an invariant may state that the principals involved in a protocol are honest, meaning all entities carry out the protocol steps as specified. Using such an approach, protocol steps can be combined as long as the two protocol steps do not violate the invariants of each other. Protocol proofs take the form $\theta[P]_x\phi$, which means after x executes actions P starting from a state where θ is true, ϕ will be true in the resulting state. SPCL uses a number of predicates.

TERMINOLOGY 2.1 COMMON TERMS IN SPCL

The interpretations for the predicates used in SPCL are given below.

- *Has(P,x):* Process *P* possesses information *x*. If information is sent encrypted with key *k* as in $\{|x|\}_k$ then recipient must be in possession of key *K* needed for decryption.
- *Send(P,m):* Process *P* sends message *m*.
- *Fresh(P,t):* The term *t* generated by *P* is fresh (recently generated).
- *Honest(Z):* Entity *Z* carries out the actions as specified by the protocol.
- *Source(m,X,M):* The only way any entity other than *X* can learn about *m* is through the set *M*.
- Temporal operator $\ominus\phi$ *:* specify ϕ has to hold in the next state.
- Temporal operator $\Diamond\phi$: specify ϕ eventually has to hold
- *After(a1,a2):* action *a2* happens after *a1*.
- *New(P,t):* P creates new term t.
- *Encrypt(P,t):* P encrypts t with its key.
- *Decrypt(P,t):* P decrypts t with its key.
- *Verify(P,t):* P decrypts t with its own private key.

DEFINITION 2.1 PREDICATE FORMULAS

Predicate Formula (ϕ) can be constructed using the BNF grammar below:

ϕ *::= a | Has(P,t) | Fresh(P,t) | Honest(N) | Contains(t1,t2) |* $\phi \wedge \phi \mid \neg\phi \mid \exists x.\phi \mid \Diamond\phi \mid \ominus\phi$

- *P and t represent a process and a term, respectively.*
- *Process here refers to an entity playing a role such as Bob playing the role of responder.*
- The symbol *m* represents a generic message each of which has a source, a destination and a protocol identifier, in addition to message contents.

Action Formula a can be constructed using the grammar below.

a :: = Send(P,m) | Receive(P,m) | New(P,t) | Decrypt(P,t) | Verify(P,t)

- *where Send, Receive, New, Decrypt and Verify are predicates defined in Terminology 2.1.*

Proof System

SPCL proof system uses axioms related to protocol actions and predicates. In this section a small collection of such axioms is described. A complete collection of axioms and sample proofs were presented in some earlier work in protocol composition [26, 61, 78].

Axioms Relating to Protocol Actions

These axioms state properties that hold immediately after executing a protocol action. In the axioms below *vn* refers to creation of new nonce *n.*

AN1 $\phi[(vn)]_x\ Has(X,n)$ says an entity generating a new nonce *n* is in possession of it.

AN2 $\phi[(vn)]_x\ Has(Y,n) \supset (Y = X)$ says if *X* generates a nonce *n*, no other entity has *n.*

Axioms Relating to Predicate Actions

These specify the relationship between predicates.

DEC $\diamond Decrypt(X, \{|n|\}k) \supset has(X,n)$ — DEC states message elements decrypted by an entity are added to its knowledge.

PROJ $has(X,(x,y)) \supset has(X,x) \wedge has(X,y)$ — PROJ states a process that possesses a tuple also possesses the tuple elements.

Generic Inference Rules

These predicates help derive new assertions from existing ones using Hoare's logic.

GR1 $$\frac{\theta[P]_x\phi \quad \theta[P]_x\psi}{\theta[P]_x\phi \wedge \psi}$$

GR2 $$\frac{\theta[P]_x\phi \quad \theta' \supset \theta \quad \phi \supset \phi'}{\theta'[P]_x\phi'}$$

Preservation Rules

Preservation rules specify predicates that are not affected by actions. For example, an entity in possession of some data element will continue to do so even after one of the actions identified earlier. This can be stated using the axiom: $Has(X,t)\ [a]_x\ Has(X,t)$

Protocol Composition Axioms

Proof of correctness using deductive systems can be extended to composed protocols when predefined formulas, known as environmental invariants (denoted by Γ), are not violated [61]. Such environmental invariants are necessary to allow reasoning about security properties during composition. The three rules described below ensure that protocols are composed only when they do not interact adversely by interfering with one another, or with the environment [61].

- **The weakening rule WR** states that a formula (ϕ) provable ($\vdash$) from invariant Γ remains provable with additional invariants.

$$\textbf{WR:}\quad \frac{\Gamma \vdash \phi}{\Gamma \cup \Gamma' \vdash \phi}$$

- **The composition rule CR1** allows protocol roles to be composed sequentially when the post-condition of the first protocol matching the precondition of the second one.

$$\textbf{CR1:}\quad \frac{\Gamma \vdash \phi_1[P]_A \phi_2 \qquad \Gamma \vdash \phi_2[P']_A \phi_3}{\Gamma \vdash \phi_1[P; P']_A \phi_3}$$

- **The composition rule CR2** states that if environmental invariant (Γ) is not violated by protocols Q and Q' then it will not be violated by the protocol obtained composing them.

$$\textbf{CR2:}\quad \frac{Q \vdash \Gamma \quad Q' \vdash \Gamma}{Q \circ Q' \vdash \Gamma}$$

Every role ρ in the composed protocol $Q \circ Q'$, (obtained combining in parallel or sequential) can be written as $\rho = \rho_1\rho_2....\rho_n$ where ρ_1, ρ_2 are basic sub-sequences of roles in Q or Q'.

Formalising Protocol Composition

The SPLC methodology [61] allows protocols to be composed in sequence or in parallel by combining the sub-sequences, subject to finding suitable environmental constraints and matching post and preconditions. The general approach for proving composed protocols can be summarised as follows:

- Prove the security properties of component protocols Q and Q' separately.
- Identify the invariants used for these proofs, and name them Γ and Γ' respectively.
- Combine protocols Q and Q' using either of the composition rules CR1 or CR2 keeping $\Gamma \cup \Gamma'$ as the environmental invariant.
- Apply the sequential composition rule CR1 when the post-condition of the first matches the precondition of the second.
- Finally, prove that the environmental invariant $\Gamma \cup \Gamma'$ holds for both protocols Q and Q'.

This protocol composition approach provides a suitable formalism for combining cryptographic schemes by making the preconditions and invariants explicit. Protocols formed through sequential composition are valid because sub-protocols are combined only when post-condition for one meets the precondition for the other. However, it does not have any inherent mechanisms for creating new types of security schemes. Although it provides a bottom-up strategy to create a more complex protocols from simpler ones, it must be combined with other top-down search strategies to reach the required goals, if it is to solve real-life problems [23]. Another drawback of this approach is that it does not include a threat model for explicitly specifying adversary capabilities, as in Strand Spaces. Hence, it cannot detect attacks when an adversary uses information gathered from one sub-protocol to subvert another, even if both sub-protocols themselves are valid. Hence, protocols devised using such an approach must be strengthened to withstand attacks such as type-flaw attacks.

2.3.3 Semantics Based Protocol Synthesis

In another approach to protocol synthesis, the Logic of Knowledge, a form of modal logic, models the global system state based on past events [156-159]. Logic of Knowledge allows inferences to be made based on past actions. However, before such inferences can be made, the roles of entities and the semantics of data elements must be standardized across the domain of interaction. In another more recent approaches e-commerce and web service protocols were synthesised using social commitments of interacting entities as the basis [62,94]. A supplier may commit to ship an item if and when the customer commits to pay for that item when delivered. The customer may in turn commit to pay on delivery if the merchant commits to replace it if it is found defective later. Viewed thus, a protocol is a series of commitments leading from the initial state to one of the final states when all commitments are either discharged or cancelled. Such an approach allows interactions to be determined at run-time using a number of primitives such as creating, discharging, cancelling or releasing commitments. This approach allows entities to make use of opportunities, thus following a less rigid path than standard protocols. For example, a customer may not require a quote from a specific merchant before placing an order, if it can be trusted to offer the best price anyway. The approach however, cannot prevent adversaries triggering new events to thwart commitments being met. Moreover, it lacks the security features necessary to initiate action against e-commerce entities failing to honour commitments.

2.3.4 Overall Evaluation of Past Synthesis Techniques

Past synthesis techniques using brute force methods such as as APG can only produce very simple two-party protocols. Such techniques cannot create realistic protocols at runtime (even with fast model checkers) as the number of possible combinations increases exponentially with the number of messages and entities. Search-based techniques provide a top-down approach that allows goals to be met by breaking them up into sub-goals that can be enforced by existing schemes. However, these techniques cannot guarantee that all goals can be decomposed into sub-goals. Furthermore, meta-heuristic search requires the weights to reflect the type of protocol environment. These difficulties make such search-based techniques unsuitable for environments where protocols must be created at runtime. Moreover, beliefs used as the basis for synthesis in BAN-based techniques cannot be presented as expicit evidence in e-commerce. The Strand Spaces approach provide a useful graph-based challenge-response mechanisms to devise new cryptographic schemes, and to prove existing schemes. However, their applicability is limited to sequential schemes as the principle of unique origin does not extend to branching behaviour. Protocol composition logic provides a formal basis for composing protocols using invariants and preconditions. However, it does not provide a mechanism for guiding protocols to meet required goals. Therefore, any practical strategy for composing protocols should combine protocol composition techniques with challenge-response mechanisms and search-based straegies. Similarly commitment based approach allows flexible protcols to be generated based on underlying semantics, though such an approach must combine messages with schemes enforcing necessary security properties to ensure entities making commitments are held accountable.

2.4 Security Protocol Attacks and Prevention Techniques

Difficulty in verifying complex security protocols (such as SET) have created a research interest in security protocol synthesis techniques [57, 59]. Adherence to certain principles during design can reduce the complexity of model checking algorithms [79]. Such techniques are also necessary if protocols are to be created dynamically to meet end-to-end security requirements of collaborating entities [14]. Moreover, the techniques presented in Section 2.3 cannot entirely prevent attacks by adversaries, as characterised by type-flaw and replay attacks. Therefore, this section surveys mechanisms devised to counter such attacks [80-84]. In addition, the general design principles derived from analysis of past protocol failures are surveyed briefly [79].

In the past most security protocols were restricted to simple key exchange protocols. E-commerce protocols, however, present many challenges as they are applied in diverse environments and devices. The trade-offs between security and performance become crucial in e-commerce, especially for resource constrained mobile and wireless devices. Therefore this section also surveys some recent research work carried out to lower the cost of security. Furthermore, many e-commerce and web service applications require security schemes to be synthesised dynamically based on specific security requirements. Many common security properties, however, have multiple interpretations, making precise specification difficult. Hence, some common security properties and their semantics are briefly surveyed in this section.

2.4.1 Requirements for Security Protocols

The lack of standard definitions for common security properties has resulted in many security protocol flaws in the past [31]. The authentication property for example, has many different notions attributed to it, each providing a different level of assurance [33]. Authentication based on aliveness requires proving that "whenever data is received by entity *B* from a trusted entity *A*, that data must have been sent by *A* to *B* in an earlier event". This assurance, however, does not guarantee that *A* initiated the data transfer recently or that there is a one-to-one correspondence between *A* and *B*. A stronger assurance amounting to *injective agreement* defines it as "if whenever *A* has completed a run apparently with *B*, then *B* has been running the protocol apparently with *A*, such that each run of *A* corresponds to a unique run of *B*" [33]. However, a stronger form of authentication requires additional overheads, which may not always be necessary. If composite properties can be decomposed into basic properties with exact meanings, past misinterpretations can be avoided. Furthermore, by using multiple properties only when necessary, security overheads can be reduced.

2.4.2 Common Attacks and Techniques for Prevention

It is not possible to exhaustively list all possible attacks as they are limited only by the ingenuity and resources of the attackers. Protocol design should therefore aim to make the cost of subverting a protocol higher than the value of the information contained therein. This section presents replay and type-flaw attacks, two of the most common attacks that can be easily mounted by an adversary. These attacks undermine protocols by resending or redirecting messages. This section also presents strategies devised in the past to make protocols resistant to such attacks. Preventing these attacks also thwarts other related attacks such as reflection attacks and man-in-the-middle attacks. This section also presents the classic man-in-the-middle attack using the attack on Needham-Schroeder public key protocol. Attacks such as denial-of-service attacks are not surveyed as the scope of this thesis does not include security components like firewalls and intrusion detection techniques necessary to prevent such attacks.

2.4.2.1 Type-Flaw Attack

A type-flaw attack occurs when a message in a protocol run is interpreted incorrectly as another type in a different protocol run. Any protocol that does not explicitly state the meaning of a message and has a message structure similar to another becomes vulnerable to such an attack. For example, the Needham-Schroeder public key protocol allows two entities to exchange nonces without the knowledge of any other entity. These nonces allow a shared session key to be created for secret communication. A trusted server is used for serving the public keys. This protocol consists of 7 messages, as shown below, and has two messages with the same structure (messages 3 and 6). The protocol is expressed using the standard notation where *A* and *B* are agent identities, *S* is the identity of the trusted server, N_A and N_B are nonces and *PK* and *SK* are public and private keys respectively.

Message 1: A→S: B	// *A* send *B*'s label to server
Message 2: S→A: $\{ PK(B), B\}_{SK(S)}$	// Server responds by sending *B*'s label and its public key
Message 3: A→B: $\{N_A, A\}_{PK(B)}$	// *A* sends its nonce and label encrypted by *B*'s public key
Message 4: B→S: *A*	// *B* send *A*'s label to server
Message 5: S→B: $\{ PK(A), A\}_{SK(S)}$	// Server responds by sending *A*'s label and its public key
Message 6 : B→A: $\{N_A, N_B, B\}_{PK(A)}$	// Nonce challenge issued to *A*
Message 7 : $A \rightarrow B: \{N_B\}_{PK(B)}$	// *A* responds to nonce challenge

An intruder *INTR* can attack this protocol by redirecting a message in one protocol run **P1** as another message in a different run of the same protocol **P2**, as shown below. It is assumed *INTR* may fake any identity using all data and keys in its possession, to sabotage a protocol. Here $INTR_X$ denotes the intruder faking the entity *X* as in messages **P1.3**, **P1.6**, **P2.3**, **P1.7**. The messages **P2.3** and **P2.4** are part of a second run of the same protocol initiated by the intruder with the agent *A*, to decrypt N_B. Note that the nonce N_B is sent in the open in message **P2.4**, which allows the intruder to respond to the nonce challenge issued in **P1.6**, in the final message **P1.7**. The possession of nonces N_A, N_B allows the intruder to correspond with *B*, pretending to be *A*.

Message P1.3: $INTR_A \rightarrow B: \{N_I, A\}_{PK(B)}$

Message P1.4: $B \rightarrow S: A$

Message P1.5: $S \rightarrow B: \{ PK(A), A\}_{SK(S)}$

Message P1.6 : $B \rightarrow INTR_A: \{N_I, N_B, B\}_{PK(A)}$ // Nonce challenge to the intruder faking the identity

Message P2.3: $INTR_{(NB,B)} \rightarrow A: \{N_I, (N_B,B)\}_{PK(A)}$ // *INTR* fakes its identity as N_B,*B* (*B* appended to N_B)

Message P2.4: $A \rightarrow S: (N_B, B)$ // N_B sent in the open allowing access by intruder

Message P1.7 : $INTR_A \rightarrow B: \{N_B\}_{PK(B)}$ // Intruder responds to nonce challenge

This intrusion cannot be detected unless a mechanism such as a tagging scheme is used which explicitly tag the data elements with the type of element used [80]. Each base element is associated with a tag that identifies common protocol elements such as agent name, nonce, public key and shared key. A tagged value of the form (Tag, Value) specifies the claimed type of value and the actual value. Using such a scheme and additional outer level tags (text) to describe the message semantics, Message 6 would be translated into:

$$(\text{“}\{|nonce,nonce,agent|\}pubkey\text{“}, \{(\text{“}nonce\text{”},NA),(\text{“}nonce\text{”},NB),(\text{“}agent\text{”},B)\}_{PK(A)})$$

The first part (text) consisting of tags *nonce*, *agent* and *pubkey* describes the semantics of the message as two nonces and an agent name encrypted with a public key, while the second part consists of tags and the actual values in encrypted form. The outer level tag in plain-form allows an honest agent to decompose the message correctly. Explicit typing of elements in the second part of the message in the inner level prevents any type-flaw attacks. Any changes to the first part of the message in plain-form can be easily detected by the recipient as a mismatch. Though the approach can be implemented with negligible additional processing cost, it increases protocol bandwidth significantly, through additional tags in encrypted and plain forms. Tagging schemes can be further simplified by leaving out the tags at the outermost level [81].

2.4.2.2 Replay Attack

In a replay attack an intruder sends an earlier message as part of another run or protocol. Replay attacks are commonly classified into internal, external and interleaving types [83]. Protocols are vulnerable to such attacks when no schemes are implemented to track protocol runs or the age of the message. A common prevention strategy involves sending additional state information as part of protocol messages, thereby making it harder for intruders to replay without being detected. State information may include a protocol identifier, a protocol run identifier, a transmission step identifier, a message sub-component identifier and primitive types of data items. In practice, however, such schemes are seldom used due to the increased bandwidth required and associated processing costs.

Overheads for replay prevention can be reduced by implicit typing [84] and hashing of full information. Implicit typing involves creating a unique encryption function for each message by making encryption a function of both the key and a constant that varies between messages and protocols. In this way a message from another protocol or other part of the same protocol can be distinguished easily, even if it has the same format. Implicit typing is more effective than other replay

prevention schemes since it requires no additional data or processing cost. However, it makes the implementation more complex, requiring protocol and agent-specific parameters for encryption/decryption. Also, it is not effective when the same protocol is replayed between the same two agents, as the encryption parameters are identical. Attaching a signed hash of full information adds little to the size of messages or processing cost [85]. Including hash of full information in an acknowledgement makes it difficult for the recipient to claim that the receipt acknowledgement was a replay; i.e., it is an acknowledgement for an earlier message received which happened to have the same message contents. Similarly, hash chaining can also be used efficiently to guarantee non-repudiation of data origin in multicast traffic where the multicast groups can be very large. [86].

2.4.3 Protocol Design Guidelines To Prevent Common Attacks

Many researchers proposed specific "dos" and "don'ts" for protocol design, based on features that caused past protocol errors [79, 87]. Even if these design techniques cannot guarantee correctness, the construction process make formal verification easier, as design techniques and formal proofs are generally complementary [79, 88]. This section describes these guidelines.

2.4.3.1 Principle of Full Information

Some designers have inadvertently introduced errors while attempting to optimise security protocols. As an example, consider the authentication protocol created by Woo and Lam that relies on key translations by a trusted server for authentication. Key translations are commonly used with symmetric keys where the server shares keys with a number of principals. The stated aim of the protocol is that at the end of the run the initiator of the protocol claiming to be *P*, is actually the same principal (*P*). The protocol below uses the standard notation for nonces (*n*), symmetric keys (K_{PA},K_{QA}) and agents (*P*,*Q*, *A*).

1. P→Q: *"I am P"*
2. Q→P: *n*
3. P→Q: $\{n\}K_{PA}$
4. Q→A: $\{P,\{n\}K_{PA}\}K_{QA}$
5. A→Q: $\{n\}K_{QA}$

This simple protocol commences with the initiating agent *P* identifying itself to another agent *Q* that then responds with a challenge nonce, which *P* must encrypt with a symmetric key it shares with trusted server *A*. *Q* then forwards the message response together with the identity of the initiating agent (*P*) to the server after encrypting it with the symmetric key it shares with *A*. *Q*'s receipt of the original nonce *n* from the server encrypted with the shared symmetric key is then taken as proof of *P*' s identity, as the trusted server is involved in a key translation. The protocol was later found to be incorrect [21], as it becomes prone to attack when the responder (*Q*) starts a second interleaving run of the same protocol with another principal. The saboteur starts a first protocol with *Q* pretending to be *P*. The saboteur cannot, however, get *A* to perform the necessary key translation required in **step 5**. The saboteur waits until *P* starts another run of the same protocol to perform the necessary key

translation using the nonce from the first protocol. This translated key is then used as the last message in the first protocol instance.

In an attempt to learn from their mistakes, Woo and Lam retraced their steps in designing the protocol and removed the redundancies. The protocol flaw was the result of removing *P*, the name of the initiator whose identity is being authenticated in the last step replacing $\{P,n\}_{K_{QA}}$ by $\{n\}_{K_{QA}}$. Incorporating the name initiator (*P*) in the encrypted part allows the responder to correctly infer the identity of the sender. They concluded that authentication protocols carrying full information gathered in every outgoing encrypted message could reduce the chance of replay attacks. A number of heuristics for safely reducing redundancies were formulated:

- Each message should contain the names of both the initiator and the responder, and the nonce of at least one participant, the recipient. The nonce allows the authentication run to be uniquely identified.
- An encrypted message intended for a particular principal can omit the name of the recipient but should include the name of the sender.
- Double encryption, where an outer encryption follows an inner one, should only be used when the message must be delivered to recipients in a specific order.

Full information makes protocols more resistant to attacks, because messages contain information specific to protocol runs. But it also makes the protocols more expensive in terms of processing and bandwidth, especially when a protocol requires multiple hops between two parties. Better trade-offs between security and processing can be achieved by only carrying full information of newly gained knowledge.

2.4.3.2 Explicitly Stating the Reasons for Cryptographic Components

Other design principles suggest the use of cryptographic elements be made explicit as a way to reduce common attacks and protocol misinterpretations [87]. These are summarised below.

- Include the identity of a principal if it makes the meaning of the message clearer. Messages that contain sender and receiver names encrypted with appropriate signatures are less prone to replay attacks.
- Protocol trust assumptions must be made explicit (such as *S* is the trusted key server).
- By making explicit why each encryption is used (i.e., for secrecy, authentication, integrity, non-repudiation or other non-security reasons) unnecessary encryptions can be removed, making the protocols more efficient.
- The recipient of an encrypted message should not assume that the sender knows the content of a message, unless the sender has signed it before encrypting it for secrecy.
- State the reason for use of nonces, i.e., binding (association) or for temporal ordering.

- When protocol messages are received, message terms are interpreted on the basis of the protocol and the step within the protocol. Thus, the protocol used, the run and the message order must be made clear to the receiver.
- The protocol designer should make the trust assumptions clear. For example, which entities (such as a trusted server) are allowed to generate new keys must be made explicit.

These principles enforce a disciplined approach that requires designers to state underlying assumptions clearly and reason out why a specific cryptographic component is used. It also raises awareness of some of the common pitfalls in past attempts to make protocols efficient. However, these general guidelines are neither sufficient nor strictly necessary for the creation of secure protocols as they provide no specific strategy to create a protocol, nor any means to identify when parts of protocol elements are redundant. The next section describes one such approach using a challenge-response mechanism, noting that most security protocol goals such as authentication and integrity have a one-to-one relationship between senders and receivers [47].

2.4.4 Summary of Common Protocol Attacks

Many of the past protocol flaws surveyed were the result of misinterpretation of security properties themselves. The authentication property, for example, has multiple interpretations, providing different levels of assurances [33]. If security requirements for protocols are to be set aggregating the requirements of all interacting parties, the meanings of basic and composite properties must be standardised [31,33]. Furthermore, the security schemes used for enforcing these properties must be provably secure. Security protocols are especially vulnerable to many forms of subtle attacks. Though it is not possible to list all possible types of attack, they can be classified into common forms of attack such as replay, type-flaw and denial-of-service attacks. Such classification allows preventive measures to be designed. A number of design guidelines that can prevent common attacks were surveyed [79, 87]. Many of the principles intended for manual design are equally applicable when synthesising protocols, though some may cause high performance degradation. Others are only applicable to key-exchange protocols, where data size, as well as the number of entities and messages is limited [87]. Designing e-commerce protocols provides many additional challenges that require combining security with other considerations such as fair exchange in transactions.

A survey of past approaches reveals that formal methods have played a vital role in verification and synthesis of security protocols [47,65,67,75]. However, underlying costs were often overlooked. Although security is considered vital in e-commerce, it cannot be considered in isolation. Increasing use of security protocols in bandwidth limited networks and resource constrained devices makes security performance trade-offs necessary. Security protocols synthesised on the basis of security goals alone may render low-valued transactions unviable. The security strength needed for a protocol should therefore be made to reflect the resources available to the potential adversary, the data value and the resulting commitments.

2.5 Accountability in E-Commerce Protocols

Many of the past protocol synthesis attempts were based on beliefs about protocol actions. Beliefs however are inadequate in e-commerce as beliefs cannot be transferred to a third party without explicit evidence. To circumvent this problem, accountability logic was devised [30] to verify whether an action can be associated with a specific party. This logic is similar to beliefs based logic where new inferences are made based on assumptions and postulates. Accountability is defined as the property whereby association of a unique originator with an object or action can be proved to a third party. In this framework proof of a statement *x* is something which convinces another principal of statement *x*. This framework allows proofs to be made to any third party (strong proof) or to a specific party (weak proof). The beliefs can be represented as form of weak proof where a party can prove a statement only to itself.

Constructs used in this framework [30] include:

Statement: "*A* says *x"*
This construct makes *A* accountable for the statement *x* and anything implied by *x*.

Weak Proof: "*A CanProve x to B*"
This construct says *A* has the capacity to prove *x* to *B* without revealing any secret *y* known to *A* ($y \neq x$).

Strong Proof: "*A CanProve x*"
This construct says *A* can prove the statement *x* to any principal *B*.

Trust: "*A is TrustedOn x*"
This construct says *A* has the authority to endorse statement *x* and is liable for making that statement

Postulates used in Proving

Postulates are presented in the form $\frac{P;Q}{R}$

This constructs means if *P* and *Q* hold simultaneously then statement *R* also holds. Here the symbol ";" is used to indicate conjunction of constructs and not temporal ordering. The symbol $\wedge$ is used for conjunction of statements such as *x* and *y*.

Some common postulates include:

Conj: $\frac{A\ CanProve\ x;\ \ A\ CanProve\ y}{A\ CanProve\ (x\ \wedge\ y)}$

This postulate says if *A* can prove *x* and *A* can prove *y* holds separately, then *A* can prove they hold together.

Inf: $\dfrac{A\ CanProve\ x;\ \ x \Rightarrow y}{A\ CanProve\ y}$

This postulate says if *A* can prove *x* holds, and if *x* implies *y*, then *A* can prove *y* holds.

Relationship to Belief: (*A Believes x*) ⇔ (*A CanProve x* to *A*)

This postulate says *A* believes *x* if and only if *A* can prove to itself that *x* holds.

Trust: $\dfrac{A\ CanProve\ (B\ says\ x);\ \ A\ CanProve\ (B\ IsTrustedOn\ x)}{A\ CanProve\ x}$

This postulate says if *A* can prove *B* (which is an authority on *x*) says *x*, then *A* can prove *x* holds.

The main benefit of this framework is that it allows existing protocols to be analysed for their accountability properties. It also allows the right level of evidences to be incorporated depending on whether the proofs are required for a specific party or to the public. Furthermore, this framework allows beliefs to be represented as a weaker form of accountability. The main limitation of this approach is that it is applicable only to those protocols for which accountability is essential. Moreover it does not allow other security properties such as secrecy to be analysed. The framework was used mainly to analyse existing protocols; it must be combined with other techniques if end-to-end security protocols are to be synthesised.

2.6 Performance Trade-offs in Security Protocols

Widespread use of security mechanisms has attracted research interest on the security/performance balance [34]. This is mainly because the viability of security mechanisms in resource constrained devices and networks depends on protocols that allow the right trade-offs between security and performance [45, 51, 89]. Inherent resource constraints include limited battery power, processing capability and memory. Attempts to achieve such trade-offs include selection of cryptographic elements with specific performance characteristics and varying the parameters, such as key lengths and encryption block-sizes for those cryptographic elements [52]. Such trade-offs are possible if the security strength of protocols is measured in terms of the underlying security strength of cryptographic elements [46]. Security strength reflects the processing effort required by an adversary to break the underlying mechanism. Security overheads can also be reduced by classifying schemes on the basis of the exact security properties. Use of signcryption, for example, avoids the need to use different keys for authentication and confidentiality [10, 13]. Others have proposed adaptive [49] protocols whereby

security levels in a protocol are varied in response to changes in network and system resources [90, 91]. Such trade-offs can be made dynamically only if distinct security schemes that are provably secure are devised to provide the required security levels.

Most formal techniques for analysing protocols simplify the models by abstracting away items that are not essential for security. However, oversimplification of protocols, such as SET used in e-commerce, makes it hard to predict performance [12]. The performance of protocols can be predicted more accurately if cryptographic elements and algorithms with known security strength/processing cost are used. Ideally, security protocols must be designed using cryptographic elements with similar security strength, as the weakest link determines overall security strength. Experiments measuring throughput of the SSL protocol show that performance can vary by a factor of five depending on the security strength of underlying elements [49]. The security strength used should reflect the resources possessed by the opponents, with threats from government requiring the highest protection, followed by corporations and individuals [85]. The time required to break the RC-4 stream ciphers used by wireless protocols WPA and WEP using a Pentium IV 3 GHz CPU can vary from 30 minutes to 5 x 10^{23} years, depending on key length. By classifying elements and algorithms that produce similar security strength better trade-offs between security and performance can be achieved for different platforms.

The literature review of security problems clearly reveals the need to consider the trade-offs between security and cost when designing new protocols. Security overheads can be reduced if provably secure schemes meet exact security needs. Security protocols can be made to match performance constraints if the underlying elements that provide the right security strength (high, medium, low) can be selected dynamically. While synthesis of security protocols has attracted much interest recently [23, 24, 60, 61] there has been little work done in finding the security performance trade-offs.

2.7 Application of Past Logics/Techniques in this Thesis

Past protocol synthesis and verification logics/techniques provide the mathematical foundations for the work done in this thesis. Logics and techniques used in this thesis include Secure Protocol Composition Logic (SPCL), Strand Spaces and Accountability Logic.

- Chapter 3 makes the assumption all intermediaries remain trustworthy for any kind of transaction. Schemes enforcing fine-grained security properties (in Chapter 3) are created by combining challenge-response mechanisms with secure protocol composition logic (SPCL) presented in this chapter. The validity of extended schemes derived for multiple recipients (in Chapter 3) are proved using Strand Spaces based techniques presented in this chapter.

- Chapter 4 assumes a more realistic trust model where trust relationships are allowed to vary over time. Such a model requires trusted paths in the form of spanning trees to be formed through one or more category specific endorsement intermediaries at run time, based on criteria specified. A spanning tree of a graph is a sub-graph which is a tree (no cycles) that connects all the vertices in the graph.

- End-to-end schemes devised in chapter 5 keep intermediaries along the trusted paths accountable for their endorsements. These schemes are derived by transferring and discharging proof obligations along the trusted paths (in Chapter 5). The validity of these end-to-end schemes are proved using Secure Protocol Composition Logic (SPCL) and Accountability Logic presented in this chapter.

The Table 2.2 below summarises how the logics presented are used in subsequent chapters.

Logic	**Chapter**	**Purpose**
Strand Spaces	3	Proving validity of schemes for multiple recipients
SPCL	3,5	Proving validity of composition schemes
Accountability Logic	5	Reasoning about proof obligations

Table 2.2 Usage of Logics Presented in Subsequent Chapters

Chapter 3: Protocol Generation using Proven Schemes (PGPS)

End-to-end protocols such as SET [6] were hand crafted to provide the needed security assurances for e-commerce transactions involving a merchant, a customer and a payment gateway. Their applicability, however, is limited, being designed for specific configurations and type of transactions. PGPS circumvents this problem by synthesising protocols based on requirements.

3.1 What is PGPS (Protocol Generation using Proven Schemes)?

PGPS is a framework which allows protocols to be synthesised dynamically meeting the end-to-end security requirements of the interacting parties. Thus, the set of security properties for any message path is set by aggregating the security requirements of constituent data elements at their end points. Basic properties in PGPS such as data authentication and receiver nonrepudiation provide security assurances to either the data originator or the data recipient. Composite properties formed by combining basic properties may provide assurances to both data originator and recipient but in specific order. The order in which individual properties must be asserted is standardized in PGPS reflecting common security needs in e-commerce. For example, by specifying that the composite property enforcing data recency and receiver non-repudiation must assert recency first, the originator gets the non-repudiable assurance that data was received and that it was received recently, while recipient gets the assurance that data was recent. PGPS thus helps to avoid many of the past protocol flaws resulting from lack of standard definition for common security properties [31]. Furthermore, standardized properties allow protocol end-to-end security requirements to be expressed as a function of security needs of interacting entities.

Security schemes synthesised for e-commerce must meet end-to-end security needs as business rules may require data to be routed through third parties. Such business rules may dictate the order in which data elements must be delivered or the way they must be grouped together. Data elements that are grouped together provide implicit evidence to recipients that they are part of the same transaction. For example, a payment order can be associated with the purchase order if they are received together. Delivering data in a specified order prevents certain recipients from gaining undue advantage, and allows recipients to receive data indirectly through trusted third parties. Fair exchange schemes prevent undue advantage to data recipient (which may fail to acknowledge) by releasing the data in stages subject to receipt of partial acknowledgement. Unlike any previous synthesis attempts, PGPS allows such business constraints to be combined with security requirements.

PGPS uses two different techniques for proving end-to-end security: one using only direct evidence from originator and recipient while the other combines direct evidence with indirect evidence from intermediaries. The first technique uses a PGPS algorithm derived by interleaving proven two-party

schemes. The second technique relies on sequential composition of two-party schemes. Use of these schemes helps avoid the need for extensive model checking and allow protocols to be synthesised at runtime. The basic two-party schemes devised are proved using SPCL. The validity of interleaved schemes are proved using Strand Spaces. The validity of sequentially composed schemes follows directly from the compositional logic used. Additional evidence incorporated into the PGPS schemes help resist common forms of attacks, such as replay and type-flaw. The sequential compositional technique presented in this chapter (forming the Fine-grained Security Layer in Table 1.2) is extended in Chapter 5 to synthesise end-to-end protocols through trusted endorsement intermediaries (End-to-End Security Layer in Table 1.2). The paths through trusted intermediaries are selected based on the endorsement trust model presented in Chapter 4 (Trust Establishment Layer in Table 1.2).

If protocols are to be synthesised for different environments dynamically, the underlying framework must allow trade-offs between security and performance. Lack of such flexibility in standard SSL protocols led to development of many variations that work within performance bottlenecks imposed by underlying communication mediums and resource-limited devices [50-52, 92]. PGPS facilitates such trade-offs by modelling the cost of protocols as a function of underlying security strength. Such an approach allows optimal security strength to be set as a function of performance bottlenecks.

3.2 Motivation: End-to-End Security for E-Commerce Protocols

Point-to-point protocols such as TLS are designed to exchange data securely between two entities providing common security guarantees. E-commerce transactions however, require data to be despatched to multiple recipients with different security properties and in specific order. TLS, though efficient and widely used cannot provide such end-to-end security guarantees [9]. For example, a customer may send a message with order and credit card details to both a merchant and a bank with access to order details restricted to the merchant and access to credit card details restricted to the bank. Enforcing such business specific security guarantees requires specially designed end-to-end protocols such as SET. TLS protocols can provide confidentiality, data authentication and data integrity assurances. However, it does not provide receiver non-repudiation which is vital for e-commerce. Moreover, e-commerce protocols must also incorporate other properties such as fair exchange to ensure no party gains an undue advantage, in case a protocol run is terminated midway.

Security requirements in e-commerce reflect the overall perception of interacting entities about the type, value and risks associated with transactions. Each transaction may involve exchange of one or more data elements either directly or through intermediaries. Business rules in e-commerce may require specific data elements to be kept hidden, grouped together, linked or delivered in a specific order. End-to-end security provides such guarantees even when data elements are sent through one or more intermediaries. This section uses the SET purchase protocol as an example to analyse how end-to-end security requirements are met. Section 3.2.1 gives an overview of the SET protocol. Section

3.2.2 analyses SET Purchase Protocol. Section 3.2.3 presents limitations of SET purchase protocol and outlines the proposed solution PGPS. Section 3.2.4 presents the requirements based on SET purchase protocol. Section 3.2.5 presents PGPS version of the protocol meeting the same requirements. Section 3.2.6 gives an overview of PGPS. Section 3.2.7 outlines PGPS research contributions.

3.2.1 SET Protocol Overview

SET purchase protocols are designed to provide all the common security properties needed for e-commerce listed in Table 1.1. The SET purchase protocol was designed to provide such end-to-end guarantees for customers and merchants by requiring them to register with a centralized certification authority. Other SET protocols include customer registration, merchant registration, payment authorization and payment capture. The SET protocol family was jointly developed by MasterCard and Visa for transactions between three main parties, namely the customer, merchant and payment gateway [9].

The SET purchase protocol involves three components:

- Merchant component that handles purchase requests.
- Payment Gateway component, a trusted financial intermediary that verifies payment information before forwarding to other financial institutions.
- Customer holding the credit-card details that initiates a purchase. The customer shares the order information with the merchant but this is not revealed to payment gateway. Similarly payment information is shared with the payment gateway but not revealed to the customer.

The main novelty of SET is the dual signature used in the purchase protocol to implement partial sharing of information while allowing all parties to verify they are handling the same transaction. Both the merchant and the payment gateway receive a hash of hidden information signed by the customer, allowing them to verify whether they are dealing with the same order and payment information. This mechanism ensures the merchant does not have access to payment information and the payment gateway does not have access to the order information, thus meeting the end-to-end requirements.

A simplified six-step purchase protocol used in verifying the SET protocol [6] is presented below. This version omits many of the complex interactions permitted in the SET protocol and simplifies some of the steps to facilitate analysis. The notation used is kept consistent with earlier work [6] which is described briefly in this section.

ID representing entity *Y*	ID_Y
Y signing by its private key	$Sign_{priSK}Y$
Encrypting with public key of entity *Y*	$Crypt_{pubEK}Y$
Nonce for freshness challenge issued by entity *Y*	$FRESH\_Y$
Nonce challenge	XID

Step 1. Purchase Initialization Request

The customer *C* sends the merchant *M* a freshness challenge *Fresh_C* and an identifier *ID_M*.

$C \rightarrow M$: ID_M, Fresh_C

Step 2. Purchase Initialization Response

The signed response from merchant *M* includes a freshness challenge *Fresh_M* and a nonce *XID*.

$M \rightarrow C : Sign_{priSK}M(ID\_M, XID, Fresh\_C, Fresh\_M)$

Step 3. Purchase Request

SET uses a dual signature (*PIDualSign*) to ensure card details and order information are not revealed to the merchant and payment gateway, respectively. First, the customer signs the concatenation of the hashes of payment instructions (*PIData*) and the order information (*OIData*). It also includes other secrets including customer primary account (*PAN*) and card details that are encrypted with payment gateway's public key. This information together with order Information (*OIData*) and the hash of the payment instructions (*PIData*) are then sent to the merchant.

$C \rightarrow M$: PIDualSign, OIData, Hash(PIData)

Here, *C* has computed

HOD = Hash(OrderDesc, PurchAmt)

PIHead = ID_M, XID,HOD, PurchAmt,M, Hash(XID, CardSecret)

OIData = XID, Fresh_C,HOD, Fresh_M

PIData = PIHead,PAN, PANSecret

$PIDualSign = Sign_{priSK}C(Hash(PIData), Hash(OIData))$,

$Crypt_{pubEK}$ P (PIHead, Hash(OIData),PAN,PANsecret)

Step 4. Authorization Request

Before requesting authorization from a payment gateway, the merchant verifies the dual signature using the hash of payment instructions and the order information. The payment instructions are then combined with transaction identifiers and the hash of the Order Information, before being signed and encrypted using the payment gateway's public key.

$M \rightarrow P : Crypt_{pubEK}$ P $(Sign_{priSK}M(ID\_M, XID, Hash(OIData),HOD, PIDualSign))$

Step 5. Authorization Response

Before responding, the payment gateway verifies the dual signature using the hash from the order information. It also checks that the customer and merchant agree on the order description and purchase amount (*purchAmt*) by comparing their hash values. Finally it verifies the validity of the customer's secret account information. If satisfied, a signed authorization response containing the transaction identifier, hash of purchase amount and an authorization code (*authCode*) is sent to the merchant.

$P \rightarrow M : Crypt_{pubEK}M(Sign_{priSK}$ P (ID_M, XID, Hash(PurchAmt), authCode))

Step 6. Purchase Response

The merchant now responds with a signed message to the customer containing the hash of purchase amount.

$M \rightarrow C : Crypt_{pubEK}C(Sign_{priSK}M(ID\_M, XID, Hash(PurchAmt), authCode))$

3.2.2 SET Purchase Protocol Analysis

SET security protocol was designed to meet both business and security related requirements of a merchant, customer and a payment gateway. Business requirements include grouping and sequencing constraints. Order and payment information from the customer are despatched together to assure that they are part of the same transaction. The public/private keys used ensures customer and payment gateway exchange data through the merchant. The presence of hashed data in the customer signed part assures the merchant that data was not tampered with during transit. The presence of a merchant ID and a merchant generated nonce assures the merchant that data elements were sent authenticated and despatched recently (not replayed).

The presence of credit card details only in the part encrypted with the public key of payment gateway in the dual signature prevents the merchant accessing them. Similarly, including only the hashed order information in step 4 prevents the payment gateway from accessing order details. Also, step 4 allows the merchant to authenticate the order details as it incorporates the merchant ID in the part signed by the merchant. The presence of the customer signed hash of payment information assures the payment gateway that these data have not been tampered with. Data signed by the payment gateway in steps 5 and 6 (authorization response and purchase response) prove the authenticity and integrity of authorization data from payment gateway while the signing by the private key of merchant and customer ensure their confidentiality is maintained. Note the protocol ensures the authorisation code must be despatched to the merchant before the customer. The security properties met by the SET purchase protocol for various data elements is summarised in Table 3.1 while grouping and sequencing constraints on data elements are summarised in Table 3.2.

Data Element	Secrecy	Originator Requirements	Recipient 1 Requirements	Recipient 2 Requirements	Evidence
Order Information (OI)	C,M	C: - EA	M: A,R,DI		Direct
Payment Information (PI)	C,PG	C: -	M: A, R, DI	PG: A, DI	Direct/Indirect
Merchant Information (MI)		M: -	PG: A, DI		Direct
Authorisation	C,M,PG	PG: -	M: A, DI	C: A, DI	Direct

Table 3.1 Data Elements and their Security Requirements

Labels for entities: Customer - **C** Merchant - **M** Payment Gateway - **PG**

A – Authentication DI – Data Integrity R – Recency

EA – Entity Authentication

Data	Data Elements	Origin	Destinations	Previous
Purchase Request	Order Information (OI), Payment Information (PI)	Customer	Merchant	
Authorise Request	Merchant Information (MI), Payment Information (PI)	Merchant	Payment Gateway	Purchase Request
Authorise Response	Authorisation	Payment Gateway	Merchant, Customer	Authorise Request

Table 3.2 Grouping and Sequencing Requirements

SET Protocol Limitations

E-commerce protocols impose many business related constraints which require limiting access to data elements sent through intermediaries, such as credit card details or order information. Although SET protocol with its end-to-end guarantees is better suited for e-commerce than transport layer protocols such as SSL, the lack of verification mechanisms has limited its use [11]. SET allows for many different variations including frequent flyer points, split payments and payment by instalment with documentation running over several hundreds of pages [13]. Many of the past verification techniques were designed to prove the validity of simple key distribution protocols with a limited number of entities and data elements. The high cost of novel constructs in SET such as dual signatures (which involve multiple public key operations) also limit its extensibility.

3.2.3 The Proposed Solution

Unlike SET which assumes a standard configuration with a single merchant and a customer, the proposed solution aims to allow flexible configurations involving any number of e-commerce entities and message elements. Furthermore, such transaction may incorporate features such as fair exchange, which are not included in standard SET. The infeasibility of crafting and proving protocols for every possible configuration and properties makes a compositional approach to protocol synthesis necessary. Moreover, standard protocols such as SET are not designed to provide the right level of security reflecting the value of data and existing relationships. Right security level of security provides the specified level of security, i.e., no less and no more. A specific security guarantee may not be necessary when data despatched has no inherent value or when peer entities trust each other. A custom designed protocol on the other hand can be made to provide the right level of security, thus allowing better trade-offs between security and performance.

Security protocols with the right level of security can be created if entities are allowed to specify the security properties and business related constraints applicable. Such an approach requires defining the basic and composite security properties necessary for e-commerce. Provably secure schemes must be devised for all basic and composite properties. These schemes must be made extensible to multiple parties as e-commerce requires data to be sent to multiple recipients in specific order. PGPS provides such an approach, where security schemes are synthesised to meet the security and business related constraints of interacting entities.

3.2.4 SET Purchase Protocol Specifications in PGPS

The PGPS protocol specifications in this section are derived directly from the SET protocol requirements stated in the previous section. Based on the grouping, sequencing and dependency constraints in in Table 3.2, three data exchanges are involved, namely, ***Purchase_Request***, ***Authorise_Request*** and ***Authorise_Response*** with specific security properties as shown in Figure 3.1. The secrecy requirements in PGPS are handled at the granularity of data element while all other security properties are determined at the granularity of data (combination of all data elements) by summing up security requirements for each data element. For example, the data exchange containing order and payment information sent from the customer to the merchant requires the security properties authentication, integrity, secrecy, entity authentication and recency. Table 3.3 shows the data, its origin, destinations, data dependency (if any) and their security requirements other than secrecy (which is enforced at data element level). These security requirements are described in Table 1.1. For data despatched to multiple recipients, such as ***Authorise_Response,*** the order of delivery can be specified explicitly (Merchant before Customer). Table 3.4 shows the secrecy requirements at data element level.

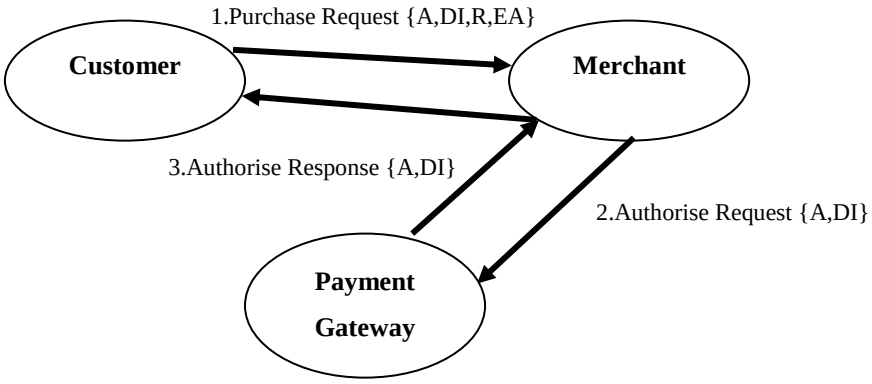


Figure 3.1 Purchase Protocol Specification Made up of Data and Their Security Properties

Data	Data Elements	Origin	Destinations	Dependency	Security Properties
Purchase Request	Order Information (OI), Payment Information (PI)	Customer	Merchant		A,DI,R,EA
Authorise Request	Merchant Information (MI), Payment Information (PI)	Merchant	Payment Gateway	Purchase Request	A,DI
Authorise Response	Authorisation	Payment Gateway	Merchant, Customer	Authorise Request	A,DI

Table 3.3 Entity Labels, Data Elements and Security Requirements

Data Element	Origin	Encryption
Purchase Information	Customer	GrpKey(Customer,Merchant)
Authorise Request	Merchant	GrpKey(Customer,Payment-Gateway)
Authorise Response	Payment Gateway	GrpKey(Merchant, Customer, Payment-Gateway)

Table 3.4 Data Exchange Details

3.2.5 Equivalent SET Purchase Protocol Composed using PGPS Technique

PGPS generates protocols combining schemes for two-party and multiple recipients, which are verified using SPCL and Strand Spaces. The security requirements along the data path are set by aggregating requirements at the end points, as described below.

- All secret data elements are encrypted at the source with a symmetric group key before performing any other operations. Secrecy group of any derived element is set to union of all base element secrecy groups. This ensures secrecy requirements are not violated.

- If entity authentication property is prescribed it is enforced in isolation without combining it with schemes for other properties. This ensures an entity itself is authenticated before any data is despatched to it.

- Data security levels are computed by aggregating the requirements for all data elements specified by the originator and the recipient.

- When data is sent to multiple recipients, PGPS schemes for multiple recipients are used to deliver data in the required order.

Table 3.5 shows the protocol generated using PGPS technique for the SET purchase protocol specification. Steps and one and two are used by customer to authenticate merchant. Steps three and four are used to get a nonce challenge from merchant to prove recency. Step five is used to send a purchase request with properties *{A,R,DI}*. The purchase request from customer includes payment information (PI) and order information (*OI*). These data elements are encrypted with group keys $\_{CP}$ (shared by Customer and Payment Gateway) and $\_{CM}$ (shared by Customer and Merchant) respectively to preserve the secrecy requirements. Steps six sends data elements payment information (*PI*) and merchant information (*MI*) with security properties authentication and data integrity. The data elements *PI* is encrypted with group keys $\_{CP}$ (shared between customer and payment gateway) and *MI* with $\_{MP}$ (shared between merchant and payment gateway) The composed scheme uses two steps. Similarly, the final sub-protocol sends authorisation information (AUT_{CAB}) encrypted with the group key shared between customer, merchant and payment gateway.

Protocol Steps	Scheme	Explanation
1. Merchant_Authenticate_Request	C[EA,nC]M	Customer sends a nonce challenge nC to Merchant to authenticate entity.
2. Merchant_Authenticate- Response	M[{nC}pri_M]C	Merchant encrypts nC by its private key
3. Recency_Challenge_Request	C[Re]M	Customer prompts merchant for nonce challenge to prove recency.
4. Recency_Challenge_Response	M[nM]C	Merchant responds with nonce challenge
5. Purchase_Request	C[PI_CP&OI_CM,{h(PI_CP&OI_CM), nM,C,M}pri_C]M	Customer forwards a signed digest containing the hash of data together with the recent nonce, and originator and recipient labels to prove *{A,R,DI}*. The data contains payment information shared with payment gateway and order information shared with merchant.
6. Authorise_Request	M[PI_CP&MI_MP, {h(PI_CP&MI_MP),M,P}pri_M]P	Customer forwards digest signed by merchant containing, hash of data and originator and recipient labels to prove *{A,R,DI}*. The The data contains payment information shared between customer and payment gateway and merchant information shared between merchant and payment gateway.
7. Authorise_Response-1	P[{AUT_PMC,P,(M,C)}pri_P, {h(AUT_PMC),C}pri_P]M	Payment gateway sends authorization details shared between customer, merchant and payment gateway with properties *{A,DI}* to merchant first and then to customer.
8. Authorise_Response-2	M[{AUT_PMC,P,(M,C)}pri_P, {h(AUT_PMC),C}pri_P]C	Merchant forwards the authorization details to customer with properties *{A,DI}*.

Table 3.5 PGPS Scheme Generated with SET Purchase Protocol Specifications

Comparison with the Standard SET Purchase Protocol

Figure 3.2 lists the main SET purchase protocol scheme described in Section 3.2.1 and the PGPS synthesised scheme. In both cases data and cryptographic elements are exchanged between customer (*C)*, merchant (*M)* and payment gateway (*P)*. PGPS uses two additional steps as merchant authentication in steps one and two are performed separately. The rest of the steps have a one-to-one mapping and provide a similar level of assurance.

$C \rightarrow M$: Purchase_Inititialisation_Request
$M \rightarrow C$: Purchase_Inititialisation_Response
$C \rightarrow M$: Purchase_Request
$M \rightarrow P$: Authorisation_Request
$P \rightarrow M$: Authorisation_Response
$M \rightarrow C$: Purchase_Response

Standard SET

$C \rightarrow M$: Merchant_Authenticate_Request
$M \rightarrow C$: Merchant_Authenticate- Response
$C \rightarrow M$: Recency_Challenge_Request
$M \rightarrow C$: Recency_Challenge_Response
$C \rightarrow M$: Purchase_Request
$M \rightarrow P$: Authorise_Request
$P \rightarrow M$: Authorise_Response-1
$M \rightarrow C$: Authorise_Response-2

PGPS

Figure 3.2 Standard Set and PGPS Generated Protocol Steps

PGPS and the standard SET approach vary mainly in the way secrecy is enforced. Standard SET uses dual encryption technique to enforce secrecy and to associate data elements [6] while PGPS uses

group keys to enforce secrecy. The main advantage with the PGPS approach is its flexibility; it can be made to provide additional properties such as non-repudiation for the order information. Furthermore, PGPS can provide the needed assurances to any number of entities and intermediaries at runtime by aggregating their security requirements.

3.2.6 PGPS Overview

This section briefly outlines how PGPS generates security schemes based on business related constraints and end-to-end security requirements. Business related constraints reflect the semantics of transactions, the value of data elements and current business policies. Section 3.2.6.1 presents the input to PGPS, while 3.2.6.2 the type of business constraints handled in PGPS. Section 3.2.6.3 presents PGPS end-to-end security requirements while Section 3.2.6.4 outlines the PGPS solution.

3.2.6.1 Inputs to PGPS

The inputs to PGPS include both business constraints and security related requirements. Both business constraints and security requirements reflect the semantics of knowledge elements, the role of entities and the type of transactions. The next two sections describe business constraints and security requirements in greater detail.

3.2.6.2 PGPS Business Constraints

Business related constraints specify data elements that must be grouped together and the order in which they must be exchanged. It is assumed the grouping and sequencing of data elements are carried out reflecting the semantics of transactions and data elements. The business constraints specified to PGPS include:

- Data elements that must be despatched together (grouping constraints)
- The order in which data must be delivered to recipients (sequencing constraints)

A grouping constraint (*GC*) takes the form $GC: E_S \times E_D \times DE$, where $E_S, E_D \in E$ the set of valid entities and $DE \in Pow(D)$, the set of data elements. For example, an e-commerce grouping constraints may require data elements order details (*OD*) and payment details (*PD*) to be despatched together from customer (*C*) to merchant (*M*) as in: $C \rightarrow M: \{OD,PD\}$.

A sequencing constraint (*SC*) takes the form $SC: Pow(D) \times < E_S \mid E_S \in E >$ where *D* is the set of data elements *and E* is the set of entities. For example, an e-commerce sequencing constraints may specify data elements *order details (OD)* and *payment details (PD)* from *customer (C)* to be despatched to *payment gateway (PE)* via *merchant (M)* as in: $\{OD,PD\}: C \rightarrow M \rightarrow PG$.

3.2.6.3 PGPS End-to-End Security Requirements

PGPS security requirements allow end-to-end assurances for data and entity identities to be stated. Data security requirements including which secrecy (S) and correspondence properties (A,DI,RNR,R) are stated at the granularity of data element. The security properties assigned reflect the value of transactions, the type of commitments and relationship between entities. The following general guidelines are used in PGPS:

- originator to require receiver non-repudiation if proof of despatch is required
- receiver to require authentication if originator to be held accountable
- receiver to require recency if validity of data element is time dependent
- sender to require secrecy if data element despatched is confidential

The following sections summarise the syntax of different types of security requirements.

Entity Authentication Requirement *EAR:* $E_A \times E_B$, where E_A, $E_B \in E$, the set of valid entities. For example, when *A* requires *B* to authenticate its identity and *B* requires both *C* to authenticate its identity, the entity requirements can be expressed as in: $A \rightarrow B$, $B \rightarrow C$

Data Security Requirements

Data security requirements include secrecy (S), data authentication (A), data integrity (DI), receiver non-repudiation (RNR) and data recency (R).

- **Secrecy Requirement** (*SR*) for a data element is specified by its originator by listing the set of entities granted access as in *SR:* $D_i \times E_O \times E_D$ where $D_i \in D$, the set of data elements, and $E_O \in E$ is the entity of origin and $E_D \in Pow(E)$ is the set of destination entities. For example, payment information *PI* originating in customer *C* and granted access in payment gateway *(PG)* and acquisition bank (*B*) can be specified using secrecy requirements: (*PI:C,{ PG,B}).*

- For **Non-Secrecy Properties** (*NSR*), both originator and recipients specify their requirements in the form *NSR:* $D_i \times E_j \times SREQ$ where $D_i \in D$, the set of date elements, $E_j \in E$ the set of entities and $SREQ \in Pow\{A,RNR,R,DI\}$ is the set of security properties. For example, assume data element order information (*OI*) originating in customer (*C*) is required in merchant (*M*) with properties {*R,DI,A*} and in payment gateway (*PG*) with {*DI,A*}. Also, the customer requires receiver non-repudiation. This is specified in PGPS as in: *(OI:C,{RNR}),(OI,M,{R,DI,A}), (OI,PG,{DI,A})*. Note the requirement in originator is specified first.

3.2.6.4 PGPS Solution

The Table 3.6 summarizes some of the problems faced in e-commerce security and how PGPS approach can help overcome them.

	E-Commerce Security Problems	**PGPS Solution**
1.	Security needs for interacting e-commerce entities may vary over time. Existing protocols assume security needs remain constant.	PGPS allows interacting entities to specify exact security needs for each data element, which are then aggregated to arrive at security needs for data exchanged. This required standardizing the order in which constituent properties are enforced for composite properties.
2.	E-commerce protocols are constrained by legal and business rules which may require grouping of data elements and delivery in specific order.	PGPS allows protocol path to be determined based on such constraints. The security properties along the path are set aggregating security requirements for all data elements along that path.
3.	Existing e-commerce protocols are designed for standard configuration. E-commerce and web services require protocols for different configurations.	PGPS synthesises schemes at runtime using proven schemes. Provably secure two-party schemes are first devised for basic properties, which are then combined to enforce composite properties.
4.	E-commerce protocols need end-to-end assurances as data may be sent through one or more intermediaries.	Indirect end-to-end assurances are provided by combining two-party schemes in sequence while direct assurances result from schemes piggybacking data and cryptographic elements.
5.	E-commerce protocol may face many security attacks including replay and type-flaw attacks.	Replay attacks can be prevented by specifying the recency property. Inclusion of additional terms can prevent attacks such as type-flaw.
6.	Security/performance trade-offs are vital in e-commerce as many devices/networks are resource/cost constrained.	PGPS cost model allows selection of cost-effective protocols. Security protocols can be made to work within performance and cost constraints by using lower security strength.

Table 3.6 E-Commerce Problems and PGPS Solutions

3.2.7 PGPS: Research Contribution

The main contribution of this chapter is a framework to synthesise security protocols based on business constraints, security requirements and performance trade-offs. The solution presented includes defining fine-grained security properties aggregating them considering other business constraints, devising schemes for fine-grained properties, extending them to multiple recipients in specified order and finding trade-offs with performance.

- Fine-grained security levels formed by combing basic properties and the algebraic operations defined over them allow security requirements to be aggregated. By standardizing the order in which individual security properties within composite properties are enforced, past protocol design flaws [31, 33] caused by multiple interpretations of common security properties are avoided. By specifying exact security requirement using fine-grained properties overheads caused by over-prescription of security requirements are avoided.

- Fine-grained security schemes are devised by combining challenge-response mechanisms for basic schemes [21, 47] with compositional logic [25, 61] used for safely combining them. These schemes are strengthened to resist common attacks by incorporating additional terms. It was shown these proven schemes can be combined to create protocols similar to SET purchase protocol at runtime. The flexible PGPS approach may be extended to other security properties such as privacy. Though a number of previous attempts have been made to synthesise protocols using challenge-response mechanisms or composition logic, no past attempt was made to combine these techniques to form fine-grained security schemes. Also the long elapsed times made previous approaches unsuitable for synthesis at runtime.

- A novel interleaving technique devised allows any two-party scheme to be extended to multiple recipients in specified order. The validity of extended schemes is proved using Strand Spaces. The underlying algorithm allows the creation of multiparty security schemes at run-time as the complexity of the algorithm devised is of order $O(n^2)$, where n is the number of recipients. The interleaved schemes provide a number of benefits. First, direct evidence from originator and recipients are included even though data is sent through third parties. Second, the number of protocol steps required at data source does not increase with the number of recipients (as evidences to and from recipients are combined together).

- PGPS cost model estimates protocol bandwidth and computational cost as a function of underlying cryptographic operations, data size and security levels. None of the past protocol synthesis attempts surveyed considered the security performance trade-offs. By expressing protocol costs in terms of security strength of underlying cryptographic elements (which can be varied), PGPS facilitates trade-offs between security and cost. Such trade-offs are be necessary if protocols are to be synthesised at runtime for a specific environment.

3.3 Statement of the Problem

Creating e-commerce security protocols dynamically based on entity requirements poses a number of challenges. E-commerce entities require end-to-end assurances as data must be passed through a number of third parties. E-commerce security protocols must consider business related constraints in addition to security requirements. Entity requirements can be stated precisely only when meanings of basic and composite properties are standardized. Many of the past protocol flaws surveyed in Section 2.4 were caused either by misinterpretation of security properties or common class of attacks. Past prevention techniques presented in Sections 2.4.2 and 2.4.3 looked into ways of blocking such common attacks, such as replay and type flaw attacks. Traditional protocol design process relies on verifying protocols with model checking and rectifying them until no flaws remain. However, high computational cost of model checking large protocols makes runtime verification impossible. Therefore, compositional techniques must be devised that are provably secure or at the least, reduce the possible venues for attacks. Even if attacks cannot be completely prevented mounting attacks can be made costly. However, additional features for improving security must be balanced with performance considerations for underlying networks and devices, as absolute security is infeasible. Many variations of SSL were created in the past to work within the resource constraints of wireless devices. Therefore, security features in synthesised protocols must reflect the extent of threats posed, the underlying environment and the value of transactions.

- Past attempts to synthesise protocols used a combination of challenge response mechanisms where a sequence of send and receive events incorporating cryptographic elements were used to assert specific properties. However, such properties did not allow for finer shades of security properties necessary in e-commerce such as non-repudiation of the receipt of a message received in a timely manner by combining non-repudiation and recency properties. Individual properties too must incorporate features necessary in e-commerce; for example, non-repudiation property must incorporate fair exchange (refer to Glossary).

- Each data exchange in e-commerce may involve one or more data elements as in the case of SET protocol, which combines order and payment information. The security level (such as $\{RNR,A,R\}$) to be assigned to a data exchange must therefore reflect the security requirements specified by originators and recipients of each constituent data element. If security levels for data are to be determined at run-time as a function of security requirements a suitable algebra must be devised over the set of security properties.

- If e-commerce protocols are to be synthesised from proven schemes, composite schemes must be devised to meet every combination of security properties that may be required by data originator and recipients. However, composite schemes formed directly by combining individual ones are known to subvert the goals of one another.

- E-commerce business rules may require data to be sent to multiple recipients in a specific order, as with the SET purchase protocol. Entities in such cases may require end-to-end assurances with either direct or indirect evidence. Indirect evidence relies on each intermediary along the path meeting such an assurance. For example, sending *d* with end-to-end recency from A to C indirectly via B requires enforcing recency from A to B as well as B to C. Direct method, however, requires both originator and each recipient to provide direct evidence to one another. Therefore schemes providing various combinations of security properties must be devised dynamically to provide such evidence to any number of recipients.

- Though it is not possible to prevent every form of attack, past research has shown most common attacks (such as type-flaw and replay attacks) can be prevented [80,81,84,85]. E-commerce protocols synthesised must therefore, incorporate these additional features to resist common attacks.

- Low valued transactions and resource constrained devices require protocols that can provide the right trade-offs between security and the underlying costs. A protocol may become viable, if the underlying computational and bandwidth are reduced by lowering the security strength of cryptographic elements. Such trade-offs can be facilitated if the cost of protocols can be expressed in terms of underlying cryptographic elements.

- Schemes for enforcing properties such as receiver non-repudiation can be very costly as it may require active involvement of trusted third parties.

These problems must be resolved to allow the automatic creation of realistic security protocols.

3.3.1 Research Questions

Synthesising provably correct security protocols at runtime based on user specified requirements need to address the following research questions.

- How can end-to-end security requirements be made a function of security requirements of all interacting parties?

- How can two-party schemes be devised for all combinations of properties in a provably correct way?

- How can two-party schemes be automatically extended to any number of recipients in a provably secure way?

- How can the right trade-offs between-security and cost be achieved for protocols synthesised at runtime?

3.4 Outline of the Solution

This section gives an overview of the proposed solution devised in PGPS. E-commerce security requirements were decomposed into basic security properties that can be combined (presented in Section 3.4.1). The non-repudiation property is made to incorporate fair exchange features to ensure that no party is disadvantaged when a transaction is aborted halfway. To avoid misinterpretation of composite properties the emergent behaviours (presented in Section 3.4.2) standardise the order in which individual properties are enforced. Composite security properties are made comparable by classifying all properties into hierarchical, partially ordered security levels using a lattice model (refer to Section 3.4.3). Distinct operators were defined to aggregate both secrecy and non-secrecy properties dynamically (presented in Section 3.4.4). Composition techniques were used to derive provably secure fine-grained security schemes from basic schemes (presented in Section 3.4.5.) Protocols were extended to multiple recipients using interleaving techniques (presented in Section 3.4.6).

3.4.1 Enforcing Basic Properties

Among the six basic properties (*A,RNR,R,EA,S,DI*) only the receiver non-repudiation property (*RNR*) requires the involvement of a trusted intermediary. However, the PGPS non-repudiation scheme limits intermediary involvement to exceptional cases, based on some prior work [7]. This scheme incorporates fair exchange feature whereby data is not fully released until acknowledgement is received for partial release (refer to Section 3.6.2.5 for details). Although this scheme improves performance by limiting intermediary involvement [7] it cannot fully guarantee non-repudiation unless strong assumptions are made about the recipient and the trusted intermediary. However, this scheme gathers sufficient evidence that can be presented to the trusted intermediary for recourse before sending the data in full. A stronger form of non-repudiation scheme with heavy intermediary involvement can be used instead if performance is not critical.

Any data exchanged in PGPS may carry one or more data elements. Schemes for all properties other than secrecy are enforced at the granularity of data. Secrecy property is enforced at the granularity of data element as different data elements are accessible to different entities. All secret data elements are encrypted at the source prior to despatch, allowing only entities with the appropriate group key to decipher them. No secret data elements are sent in the open.

TERMINOLOGY 3.1 EMERGENT BEHAVIOUR

Emergent behaviours are additional behaviours that emerge from the interaction of component behaviours within a composite system [93]. In PGPS, emergent behaviours result from the way the underlying schemes enforce each component of composite properties. These emergent behaviours are standardized by predefining the order in which basic properties must be enforced, to promote common understanding between interacting entities.

3.4.2 Emergent Behaviours of Composite Properties

PGPS avoids possible misinterpretations by standardising the emergent features of composite properties on the basis of their semantics. For example, when authentication or recency are combined with non-repudiation, the standard emergent behaviour requires that the non-repudiation assurance follows authentication or recency. This follows the intuition that when a recipient non-repudiates receiving data sent with authentication and recency properties, the action is assumed to have verified its authenticity/recency. Standardising emergent behaviours also allows reasoning that involves protocols and their security levels.

TERMINOLOGY 3.2 CORRESPONDENCE PROPERTY

Correspondence properties are those security properties that rely on a one-to-one mapping between a send event and a receive event. All PGPS properties other than secrecy requires such a one-to-one mapping.

Scheme Authentication property is an example of a correspondence property as every receive event must be preceded by a send event and every send event must have a matching receive event.

TERMINOLOGY 3.3 SECURITY LEVEL

Security level in PGPS is the set of correspondence properties needed to meet the security goals of message originator and recipients.

TERMINOLOGY 3.4 DATA ELEMENTS AND DATA

Data elements in PGPS are indivisible (atomic) knowledge units with distinct semantics. Data refers to a group of data elements (de_1,$de2_{2,..}$) that are transferred together.

In PGPS, the secrecy property is enforced at data element level while data authentication, data integrity, data recency and non-repudiation are enforced at message level (aggregating security properties required for all data elements forming the message).

3.4.3 PGPS Security Properties and Security Levels

Basic correspondence properties in PGPS are derived decomposing aggregate security requirements of data originator and recipients, which includes authentication, non-repudiation, recency, data integrity and entity authentication as shown in Table 3.7. PGPS security levels are defined combining these correspondence properties in a predefined order. Standardising security levels allow aggregate security requirements of data originator and recipients to be specified precisely. For example, PGPS allows a recipient to request a composite property which combines authentication with recency, thus preventing replay attacks. Such an approach also helps to avoid past protocol flaws resulting from different

interpretations attached to common properties such as authentication (refer to Section 2.4.1). Also, by defining a hierarchy of security levels PGPS makes aggregation of security properties at runtime possible. The only non-correspondence property (which has no one to one mapping) is shown in Table 3.8. Such a modular approach allows other elementary properties such as anonymity to be incorporated if necessary.

Property	**Features**
Data Authentication (A)	Guarantees data origin and destination entities
Data Integrity (DI)	Guarantees that data is not altered in transit
Recency (R)	Guarantees that data was generated/sent after a specific event
Receiver Non-Repudiation (RNR)	Provides evidence that data reached intended destination
Entity Authentication (EA)	Guarantees identity and freshness of peer entity

Table 3.7 Basic Correspondence Properties

Property	**Features**
Secrecy	Data elements kept private between specific entities

Table 3.8 Non-correspondence Properties

PGPS security levels form a lattice model as shown in Figure 3.3. Each node in the lattice represents a distinct security level (*SL*). Note security level in the lattice does not include the correspondence property entity authentication (EA) as it is not normally used in conjunction with others.

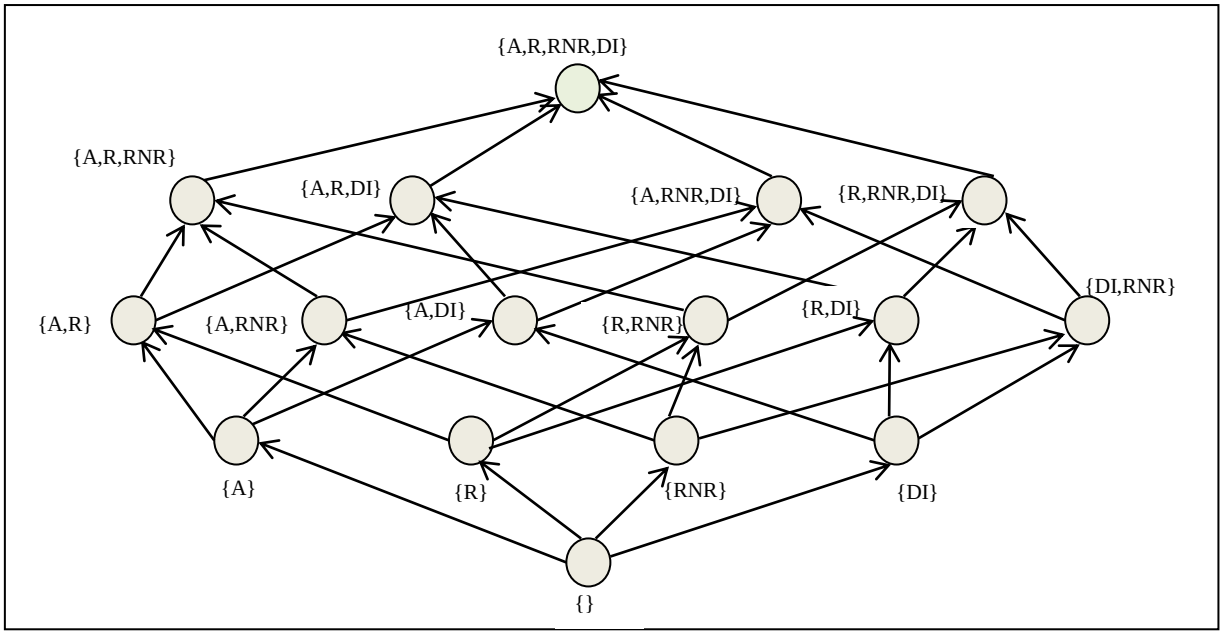


Figure 3.3 Lattice Model for PGIP Security Levels

PGPS security levels are partially ordered by set inclusion; that is for any security levels *SL1* and *SL2* in this model, *SL1* < *SL2* iff *SL1* $\subset$ *SL2*. Thus *SL1 = {A,RNR, DI}* is considered greater than *SL2 = {A,RNR}*, but *SL2 = {A,RNR}* and *SL3 ={A,DI}* are unrelated.

TERMINOLOGY 3.5 SECRECY GROUP

The Secrecy group for a (secret) data element in PGPS is the set of entities granted access to it.

3.4.4 Aggregation of Security Properties

In PGPS data the security level and data security group (defined below) are determined on the basis of their constituent elements.

The sub-protocol security level is derived by summing up the security requirements of all data elements as in : $SL(d=de_1 \cup de_2 \cup \ldots de_n) = SR(de_1) \cup SR(de_2) \cup \ldots SR(de_n)$. For example, if $SR(de_1) = \{RNR\}$ and $SR(de_2) = \{A,R\}$ then $SL(de_1 \cup de_2) = \{A,R,RNR\}$. This follows the intuition that the security level for data must meet the requirement for all its base elements.

Secrecy group (*SecGrp*) for a derived element is set as the intersection of secrecy groups for its base elements as in : $SecGrp(de_3=f(de_1,de_2)) = SecGrp(de_1) \cap SecGrp(de_2)$. This follows the intuition that if $secret_1$ is restricted to $group_1$ and $secret_2$ is known to $group_2$ then their combination must be restricted to those entities that are in both groups (intersection).

3.4.5 Methodology for Creating Fine-grained Schemes

The compositional approach used in PGPS helps to expedite the process of creating fine-grained security schemes. The five main steps used in the methodology are:

1. develop two-party schemes for all basic properties using cryptographic elements and algorithms
2. identify the assumptions and invariants necessary for these schemes, and develop the necessary proofs
3. derive schemes for multiple properties by combining basic non-conflicting schemes
4. when schemes for basic properties violate the invariants/assumptions of other schemes, devise alternative schemes (though less efficient) and repeat steps 2 – 4.
5. strengthen schemes with countermeasures to resist common attacks.

3.4.6 Extending Schemes to Multiple Recipients

Fine-grained two-party schemes are interleaved to allow multiple recipients. Strand Spaces are used to prove the validity of these interleaved schemes. The efficiency of interleaving algorithms is evaluated using experimental results by varying the number of properties and the number of recipients. These experimental results are presented in Section 3.6.6.2.

Security costs for these schemes are computed in terms of protocol bandwidth and computational overheads based on underlying digests, nonces, keys, headers and cryptographic operations introduced. The cost of enforcing secrecy is estimated at the data element level because the number of secret data elements and the entities given access to them may vary. Security overheads for all other properties are estimated at the message level as these operations are to all data elements.

3.5 The Proposed Model (PGPS)

PGPS aims to synthesise security protocols at runtime that meets the requirements of interacting parties. Such protocols are required in web services and e-commerce where entities collaborate together to meet specific goals. Section 3.5.1 explains the rationale for the PGPS model before giving an overview. Section 3.5.2 describes the model elements in greater details.

3.5.1 Rationale and Overview of PGPS Model

Dynamic collaboration between e-commerce service entities poses many challenges. Two or more service entities can collaborate to provide a composite service only if their functional and security specifications match. Past attempts to match service specifications have modelled mainly functional aspects where the permitted transition for each service is represented using finite state machines, commitments machines and AI planning techniques [1, 94, 95]. Though security is considered vital in e-commerce, lack of a mechanism to express common e-commerce security requirements made security modelling difficult [2, 15]. PGPS uses fine-grained security properties to express and aggregate security requirements of both data originator and recipient. If service entities include security levels required for permitted transitions (considering the semantics of data being exchanged), security requirements for composed services can be computed easily by aggregating them. For example, an e-commerce entity can specify time-bound non-repudiation evidence for receipt of a purchase-order message by augmenting it with the composite property combining non-repudiation and recency. Incorporating security aspects is known to increase the cost of collaboration. However, the PGPS cost model makes it possible to select the least costly protocol in terms of bandwidth and computational costs. Furthermore, the use of proven security schemes in PGPS makes it possible to meet the security needs of composed services at runtime. For example, a protocol meeting SET purchase protocol requirements was regenerated within 17 milliseconds using a 2.5 GHz laptop.

Fine-grained security levels are formed by combining common security properties needed for e-commerce including data authentication, data integrity, recency, receiver non-repudiation and entity authentication. These security properties are classified into those providing security assurances to either data originator or recipient. The secrecy property is provided at data element level allowing different recipients access to different parts of the message. Correspondence properties are enforced at message level combining security requirements of each data element. Security schemes are first devised for basic properties. Security schemes for composite properties are formed combining basic schemes when the schemes used for basic properties are not in conflict. Alternate schemes are combined when basic schemes are in conflict. These schemes are then extended to multiple recipients by interleaving the schemes. The validity of basic schemes is proved using modal logic. The validity of composite properties follows directly from SPCL. The validity of schemes for multiple recipients was proved using Strand Spaces.

3.5.2 Model Elements

Model elements consist of basic and composite properties, operators for aggregating security requirements, provably secure schemes that provide the required security assurances, functions expressing protocol costs in terms of underlying elements and scheme generators that extend the schemes to multiple recipients. The security properties include both secrecy and correspondence properties.

3.5.2.1 A Motivating Example: Interaction between Semantics and Security

This section illustrates how fine-grained security levels can be set considering the semantics of data exchanged, transaction category and the order in which sub-protocols are invoked, using a simple transaction for performing a trade. In Figure 3.4 all sub-protocols exchanged between customer and merchant including **Request** (quote), **Quote** and **Purchase-Order** are assigned different security levels. The **Request** sub-protocol involves no commitment on either the customer or the merchant and therefore requires no security assurances. The customer however, may want the **Quote** authenticated by the merchant before responding to it. The customer may also want the non-repudiation property assigned to the **Purchase-Order** to prevent the merchant denying its receipt, possibly based on an earlier favourable quote. The merchant may want assurances that the **Purchase-Order** is authentic, recent and has not been tampered with during transit, before non-repudiating its receipt. Hence, the **Purchase-Order** combines authentication, data integrity, recency and non-repudiation.

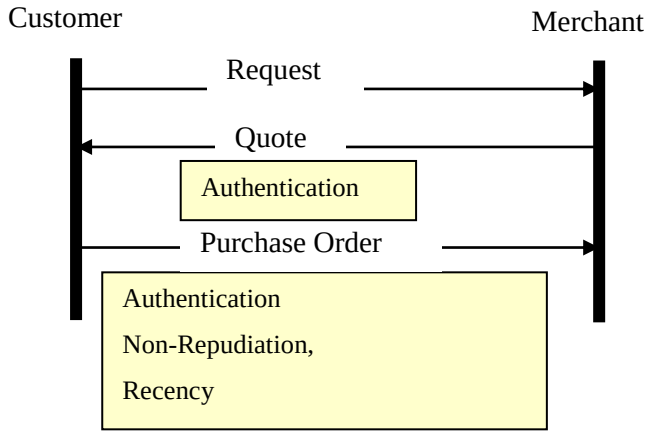


Figure 3.4 Messages using Different Combinations of Security Properties

3.5.2.2 Specifying Security Requirements

Requirements for security protocols can be classified as extensional and intensional [96]. Extensional requirements explicitly state the properties required such as authentication or data integrity. Intensional requirements specify the property in terms of the protocol itself. For example, an intensional property may state that one user responds to a specific message using a shared key. Intensional specifications are harder to verify as they require process interactions to be specified using a language such as CSP (Communicating Sequential Processes) [97]. Extensional requirements can be classified into correspondence (Terminology 3.2) and non-correspondence properties [31]. Secrecy, a

non-correspondence property, specifies restrictions on data access. Correspondence properties for common e-commerce protocols such as SET includes a combination of entity authentication, data integrity, data origin authentication, receiver non-repudiation and recency [9].

3.5.2.3 Expressing Fine-grained Levels of Security

A fine-grained security level in PGPS is formed by combining any of the basic properties for data security. The security levels (Terminology 3.3) are partially ordered using a lattice structure similar to the one used for information flow, as shown in Figure 3.3 [98]. Two unequal security properties X and Y are related (lower or higher) if there are directed edges connecting them in the lattice, either directly or indirectly. The required security level for a sub-protocol in PGPS is computed using the Least-Upper-Bound operator ($\oplus$) which takes two security properties as operands and returns the least security level greater than or equal to both (by finding their union). For example, when data *d* from *S* is sent to *R1* and *R2* through a given path *p*, the security level *SL* along that path *p* is determined based on the security requirements (*SR*) of entities *S*, *R1* and *R2* for that data as in: $SL(p,d) = SR(S,p,d) \oplus SR(R1,p,d) \oplus SR(R2,p,d)$. When data *d1*, *d2*, sent along the same path *p* are combined, the required security level is also determined using the same $\oplus$ operator as in: $SL(p,d1 \cup d2) = SL(p,d1) \oplus SL(p,d2)$. For example, if the first data requires *{A,R}* and the second data requires *{A,RNR}*, then the composite data requires *{A,R,RNR}*.

3.5.2.4 Cost of Security Protocols

PGPS expresses protocol costs in terms of underlying protocol security strength on the assumption that cryptographic elements such as key lengths, hashing algorithms, etc. can be classified according to the security strength they provide. Identifying such cryptographic elements and algorithms, however, is beyond the scope of this thesis. Security strength based costing allows security strength for synthesised protocols to be set reflecting the security/cost trade-offs for the environment.

3.5.2.5 Schemes Enforcing Secrecy

PGPS combines information flow control techniques with cryptographic mechanisms to enforce secrecy [99, 100]. Every data element in PGPS is assumed to have a single entity of origin. Furthermore, all secret elements are sent in encrypted form allowing each data element to be uniquely labelled as *secret* or *public*. This allows PGPS protocols to be type checked to prove the absence of undesirable results, such as leakage of secret elements.

Common cryptographic techniques for enforcing secrecy of data elements, include symmetric keys shared by a group of entities, multiple key ciphers [101] or a public key system. For multiple key ciphers the number of keys required is the same as the number of entities, though public/private keys operations make them computationally expensive. Public key systems produce multiple data

elements, as each secret data element must be encrypted by the public keys of all intended recipients. Thus, PGPS uses symmetric group keys for secrecy which allows a group of entities to share one or more secrets. It is assumed that all entities sharing secrets (thus known to each other) use a common symmetric group key server. Any data originator can request a group key by sending the identities of the group entities, signed by its private key. If the specified group key already exists, the server returns that key, otherwise it creates and returns a new key.

3.5.2.6 Schemes Enforcing Correspondence Properties

Enforcing correspondence properties requires message originator and recipients to possess the necessary functions and keys. All entities are assumed to share a common hashing algorithm. All entities are assumed to possess the public keys of those entities sending non-repudiable evidence of data origin or receipt, using their own private keys. All protocol entities are also assumed to possess the means to create nonces used in enforcing recency. Data received is considered recent if the message also contains a nonce that the recipient had generated recently.

Messages may also include signed entity labels which allow specific actions to be associated with entities, based on predefined conventions. For example, an entity sending a message to a specific recipient by attaching a signed label denoting the identity of the recipient in pre-specified format (possibly with some tags explicitly stating the semantics) cannot dispute later who the intended recipient for that message was. Each basic scheme provides a security guarantee to the sender or the recipient and specifies explicitly the necessary assumptions and invariants.

3.5.2.7 Scheme Generators

Scheme generators in PGPS are routines that interleave schemes in a specified order to create extended schemes that allow data to be sent to multiple recipients. Parameters passed to these routines may include source entity (*O*), destination entities (*DEST*), combination of non-secrecy properties required (*Pr*), data (*d*) and the secrecy requirements (*Sec* which is optional). The scheme generators ensure that different runs of the protocol can be distinguished by incorporating the necessary features, such as nonces. The validity of the protocols generated is proved using Strand Spaces (refer to background 2.2.1.2). The input to the protocol generator and the structure of an output protocol is listed below. An example for specific entities and properties is given later in this section.

$$pro(O, DEST, Pr, d, Sec) = {}_{X1}[C1]_{X2} \cdot {}_{X2}[C2]_{X3} \cdot {}_{X3}[C3]_{X4} \cdots$$

where *O* is originator, *DEST* is the set of destination entities, *Pr* the security properties, *d* the data, *Sec* the secrecy group for each secret element in *d*. The symbols *C1*, *C2*, *C3* are messages formed by the protocol generator routine combining encryption, decryption, hashes, nonces and data elements, based on the parameters *O*, *DEST*, *Pr*,*d* and *Sec*. For each sub-protocol hop of the form ${}_{Xi}[Ci]_{Xj}$, *xi* represents the source and *xj* represents the destination.

DEFINITION 3.1 MESSAGE TERM

The message terms (MT) are constructed using the grammar below.

$MT = S \mid D \mid E \mid N \mid F(MT) \mid (MT,MT)$

where:

Symbol	Meaning	Sample elements
S	Symbols	*Re*
D	data	*d1,d2,…*
E	entities	*O, D1,D2,…X1, X2, X3,…*
N	nonces	n_X, n_{XY}, $_Xn_Y$
F	encryption and hash functions	
	• symmetric keys	sym_{XYZ}
	• private key	pri_X
	• public key	pub_Y
	• one time key	K_{OTX}
	• hash function	h
D_{OTX}	data encrypted by one time key K_{OTX}	D_{OTA} , D_{OTB}

Table 3.9 Symbols Used and Sample Values

The nonces embedded as part of the messages could take the form n_X, $n_{XY.}$ or $_Xn_Y$ as described below. The first and second forms are used for enforcing recency and the last form for enforcing non-repudiation. When a nonce is used for recency, the notation n_X indicates origin *X*. When multiple recipients are involved, nonces used for recency can be combined. For example, n_{XY} is the result of appending the nonce generated by *X* to that generated by *Y* to assert recency. Nonces may also be used to allow one party to infer certain properties about another entity, such as the possession of certain private keys for proving identity. In such cases the notation $_Xn_Y$ is used, which indicates the nonce was generated at *X* to be consumed at *Y*. Other common terms used and some sample values are listed in Table 3.9. Functions are classified into encryption functions represented with $\{\}_{key}$ and hash functions represented with *h()*. Keys can be symmetric group keys such as sym_{XYZ},, public keys such as pub_Y, private keys such as pri_X or one-time keys such as K_{OTX}. Data encrypted by a one-time key K_{OTX} takes the form D_{OTX}. The symbols convey a predefined meaning to the peer entities such as *Re* for requesting nonce challenge to prove recency. For example, a valid scheme for passing data *d* between *A* and destinations *B* and *C*, using the composite property authentication (*A*) and recency (*R*) would be:

$$\text{pro}(A,\{B,C\},\{A,R\},d) = {}_A[Re]_B \bullet {}_B[Re,n_B]_C \bullet {}_C[n_{BC}]_A \bullet$$
$${}_A[d, \{h(d),n_{BC},A,(B,C)\}_{PriA}]_B \bullet {}_B[d,h(d),n_{BC},A,(B,C)\}_{PriB}]_C$$

In the first hop A initiates the protocol by sending the symbol *Re* (for recency) to the first recipient *B*. *B* forwards the symbol together with its own nonce (n_B) to second recipient *C*. *C* appends its own nonce (n_C) to that of A's to form (n_{BC}) and forwards it to *A*. *A* creates a signed digest consisting of data hash, the combined nonce, and labels for sender (*A*), and recipients (*B,C*) and forwards it to *B* together with the data. *B* subsequently forwards the data and hash to *C*.

3.6 PGPS Details

This section covers the underlying assumptions, the schemes for preventing common attacks, the precise definition of individual properties and cryptographic mechanisms to be used in schemes. Security protocols are usually designed with a number of assumptions, including the threat model, the level of threat posed, the existing trust between entities and the properties of cryptographic components. PGPS uses the threat model attributed to Dolev-Yao [102], which assumes the communication medium is under the control of an adversary. The protocol must be designed to withstand attempts by the adversary to redirect, fake or replay messages in the communication medium. PGPS also assumes that trusted entities are not involved in mounting an attack or leaking information to an adversary.

The basic schemes devised for the common properties are presented in Section 3.6.2. These schemes assume the availability of a public key infrastructure (PKI). PKI uses a trusted party to allow data to be exchanged securely over an insecure public network. This requires the trusted party to maintain a public key for each entity. Symmetric keys are used only for secrecy here, and it is assumed that the infrastructure includes a key server and protocols such as ELK, for distributing the symmetric group keys [103]. These schemes also assume a logical time stamping mechanism (refer to Glossary).

The security schemes devised are strengthened to withstand common attacks such as those presented in Section 3.6.1. The basic properties and underlying schemes are presented in Section 3.6.2, their proofs in Section 3.6.3. The composite properties formed combining basic properties and their combined schemes are presented in Section 3.6.4. The techniques for extending these schemes to multiple recipients and their proofs are presented in Section 3.6.5. Note all schemes devised in this chapter make perfect encryption assumptions as defined in Section 2.1.2.

3.6.1 PGPS Techniques for Preventing Common Attacks

In general it is not possible to prevent every form of attacks for three reasons. First, the number and types of attacks is not limited. Second, even if every type of attack is known, techniques used to prevent them can severely downgrade performance. Third, the cost of attacks may outweigh the potential gains for the adversary. Hence, it is sufficient in most cases to prevent common attacks that can be easily mounted by an adversary. This section describes techniques developed for preventing two of the most common forms of attack, replay and type-flaw attacks. These techniques can be extended to thwart other less common attacks such as oracle attacks. However, preventing attacks such as denial-of-service using other security mechanisms is beyond the scope of this thesis. Reasoning about protocol actions by honest entities and adversaries involved in sending, replaying or redirecting messages, require an explicit threat model. PGPS uses the Dolev-Yao threat model and the intruder capabilities presented in Section 3.6 to reason about replay and type-flaw attacks.

3.6.1.1 Preventing Replay Attacks

All PGPS schemes providing the data authentication property contain the encrypted identities of the data originator and data recipients. To avoid replay attacks at a later time, a logical time stamp (refer to Glossary) is included in the encrypted part. Thus, a sub-protocol sending data *d* from *A* to *C* must encrypt all four elements using the private key of *A*, as shown below. The hash of data is encrypted as it is computationally cheaper than encrypting the whole data, while it produces the same assurance.

$$_{A}[d,\{h(d),A,C,TS\}_{PriA}]_{C}$$

The time stamp in the signed part prevents an adversary from mounting a replay attack at a later time, thus providing authentication with assurance of a unique run. However, this logical time stamping is not strictly necessary, as either a direct (this chapter) or indirect (in Chapter 5) form of time stamping is provided as standard properties (recency or time-bound). The latter involves sending the data together with a nonce the recipient has generated recently. The presence of the intended recipient *C* in the signed part prevents an adversary intercepting the message, from forwarding it to any entity other than *C*.

Similarly, all schemes providing the data integrity property include the identity of the data originator and a hash of the data in the signed part. The sub-protocol below sends data *d* from *A* to *B* providing the data-integrity assurance.

$$_{A}[d,\{A,h(d)\}_{PriA}]_{B}$$

3.6.1.2 Preventing Type-Flaw Attacks

Protocols can be easily undermined by attacks such as man in the middle attacks if every message is deciphered solely based on message format and content. Protocols formed combining multiple sub-protocols also have been shown to open up new avenues for type-flaw attacks because of the similar message structures [81]. If, however, the current protocol state and the semantics of data elements are taken into consideration, such attacks can be easily prevented.

PGPS tagging schemes are designed to prevent type-flaw attacks (which also prevent man-in-the-middle attacks). These schemes carry an initiator-signed element that includes its security properties, the sub-protocol run and the labels of the data source and recipients, allowing it to be uniquely identified. This tagging scheme also prevents an attacker masquerading as some other entity, which would be possible if the sender were not explicitly specified. However, strengthening security by using such a scheme comes at a cost, since this scheme requires each entity to perform one additional decryption as well as additional memory space for keeping track of schemes. Tagging is made optional in PGPS, as tagged data also increases the protocol bandwidth used.

3.6.2 Security Mechanisms to Meet Semantics of Properties

This section presents security mechanisms devised for the basic security properties identified in Section 3.5.2.2. The underlying mechanisms for correspondence properties are presented in Table 3.10. Each of these schemes uses either a challenge-response mechanism or public key mechanism or a combination of these using ½, 1, 1½ or 2 cycles, where the number of cycles is defined as follows.

TERMINOLOGY 3.6: NUMBER OF CYCLES IN A PROTOCOL SCHEME

The number of cycles in a protocol scheme is the total number of send-and-receive events in the protocol originator. A half a cycle is one where there is a send event with no corresponding receive event as in the data integrity scheme presented in Section 3.6.2.3.

3.6.2.1 Data Authentication

The PGPS data authentication property provides the data recipient with the assurance that particular data, supposedly from a given entity, was actually sent by that entity and that it is the intended data destination. These assurances are enforced by incorporating the labels for the data originator, the intended recipient and the hash of data in the signed message digest. The underlying mechanism described in Table 3.10 has two main benefits. Firstly, the non-repudiable evidence from the data originator provides the recipient the assurance it is authorised to act on the data received. Secondly, it avoids the man-in-the-middle attacks where messages are redirected to unintended recipients. PGPS does not deal with man-in-the-middle attacks where messages are captured. The authentication property is enforced by using a publi $d,\{h(d),A,B\}_{PriA}$ shown in Figure 3.5.

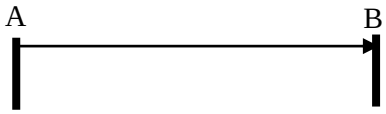


Figure 3.5 Mechanism for Authentication

PGPS allows any entity to authenticate data to multiple trusted recipients even if data is sent indirectly (to meet specific sequencing or physical requirements). Assume data from *A* is to be sent authenticated to *B* and *C*. Encrypting data with the private key of the data originator provides the necessary evidence to prove that data originated in that entity. Thus, data from *A* can be sent authenticated to both *B* and *C* (multiple recipients) as in: $_A[d,\{h(d),A,(B,C)\}_{PriA}]_B$, $_B[d,\{h(d),A,(B,C)\}_{PriA}]_C$

3.6.2.2 Entity Authentication

Entity authentication is the process where one party is assured of the identity of a second party [104]. This property is implemented with a public key based scheme in PGPS by challenging the responding entity to return a nonce sent, encrypted with its private key as shown in Figure 3.6. The underlying mechanism is described in Table 3.10.

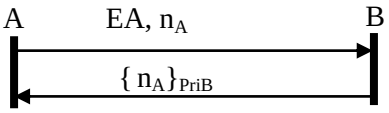


Figure 3.6 Mechanism for Entity Authentication

3.6.2.3 Data Integrity

Data integrity provides the assurance that the data has not been tampered with during transit. Any alteration to the original data can be easily detected by the recipient. A public key based scheme used for data integrity is shown in Figure 3.7. The underlying mechanism is described in Table 3.10.

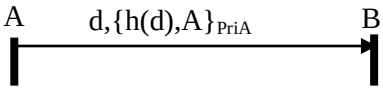


Figure 3.7 Mechanism for Data Integrity

The scheme contains only a part of the information used in data authentication, as it contains only the identity of the sender in the encrypted part. Therefore, no additional data exchange is involved when the data integrity property is combined with data authentication. The above scheme can be extended to multiple destinations (such as *B* and *C*) as in: $_{A}[d,\{A,h(d)\}_{PriA}]_{B}$. $_{B}[d,\{A,h(d)\}_{PriA}]_{C}$

3.6.2.4 Data Recency

In e-commerce the data received may not be considered valid unless it was despatched within a specific time frame. The recency property assures the age of the data received. Time stamps and nonce-based challenge-response schemes are common mechanisms for implementing recency. Time stamps with synchronised clocks can provide assurance about the time interval within which a message was generated [85], although accurately synchronising clocks for the sender and receiver can be a challenge in practice. Chapter 5 uses such a scheme, as it allows intermediaries through which a message is passed to be held accountable. A nonce-based challenge-response mechanism is used in this chapter to enforce recency, where the data is accompanied with a nonce from the recipient, as shown in Figure 3.8. The symbol *Re* is used to prompt the recipient *B* to issue a nonce challenge. By sending the data together with the nonce the recipient can be assured that the data was sent recently (after the creation of the nonce). If non-repudiable evidence is required recency can be combined with other properties.

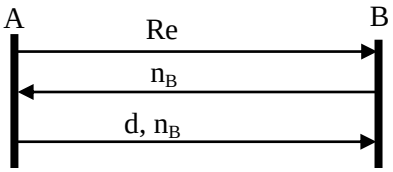


Figure 3.8 Mechanism for Data Recency

3.6.2.5 Partial Receiver Non-Repudiation

The receiver non-repudiation property requires cryptographic evidence that can be presented to an arbitrator to prove receipt of data. Unlike the other properties considered so far, this property requires data recipient to provide evidence to the sender. Fairness is considered an important goal in non-repudiation schemes as it prevents any party gaining unfair advantage through early termination [105]. A common technique for ensuring fairness is the use of a trusted third party (TTP) which delivers the key only after receipt of the non-repudiable evidence from the data recipient. However, this technique increases the traffic at the trusted third party (TTP) and the turn-around time significantly.

PGPS uses a two stage scheme based on earlier work done by Zhou and Gollman [7], where gradual release of information in first stage is combined with limited TTP involvement in second stage to gain greater efficiency, with the assumption that in most cases honest entities are compliant with protocol steps and the communication infrastructure is reliable. In such cases the TTP's involvement may be required in the second stage, and only in the case when the recipient fails to acknowledge the receipt of a key [7]. Fair exchange is achieved by sending data encrypted with a one-time key in the initial step, as shown in Figure 3.9. The one-time key k_{OTA} is sent in step three, only after the receipt of *B*'s acknowledgement consisting of the data encrypted with the one-time key, nonce ${}_{A}n_{B}$, the labels *B* and *A*, all encrypted with the private key of *B*, in step two. If the protocol is terminated after the first or second step, no party can gain an unfair advantage, as the recipient lacks the key for the cipher-text, or the sender can only prove the receipt of the cipher-text [7]. In the final step the recipient is expected to acknowledge the receipt of the one-time key. If the recipient fails to respond to step 3, the hash of d_{OTA} encrypted with the private key of *B* in step 2, can be presented as evidence to TTP (in stage 2), to initiate the necessary action to get the required evidence $\{k_{OTA}\}_{priB}$. In Section 3.6.5, this scheme is extended to multiple recipients under the assumption that a common TTP exists that can initiate corrective action when necessary.

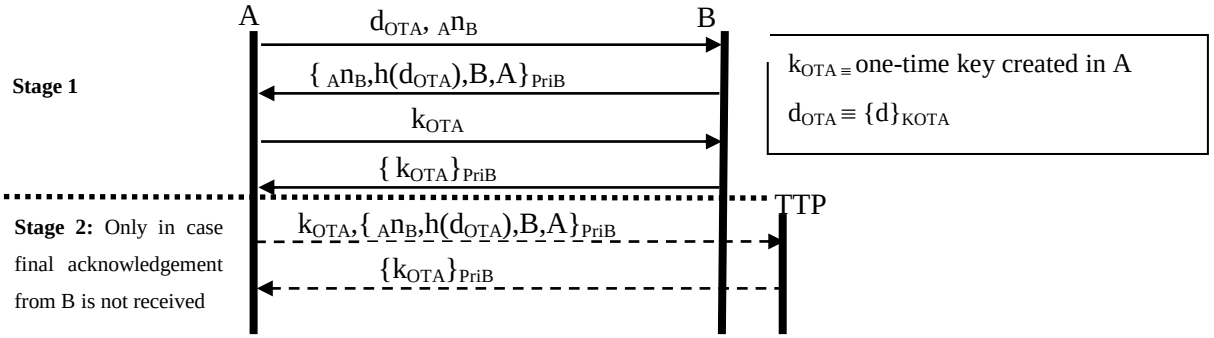


Figure 3.9 Mechanism for Receiver Non-Repudiation

Limitations

This scheme based on earlier work which claimed to be efficient because of limited TTP involvement [7], can only provide partial receiver non-repudiation in practice; the timely receipt of non-repudiable evidence cannot be guaranteed as it depends on the TTP's communication channel with the data originator and the recipient. Furthermore, a noncompliant recipient may not even respond to the trusted third party. To circumvent this problem, the framework presented in chapter 4 allows trust relationships with noncompliant entities to be severed or lowered.

3.6.2.6 Secrecy

Protocols enforcing secrecy hide information from those outside the given group. However, protocols can enforce secrecy requirements only during the data transfer phase. Other mechanisms, such as authorisation schemes limiting access to (unencrypted) secret data in storage are beyond the scope of this thesis. To enforce secrecy, secret data elements are always despatched in encrypted form; thus, only entities with the appropriate group key can access a private data element. To prevent indirect

access, the secrecy group of all derived elements in PGPS is made at least as restrictive as the secrecy group of all base elements. Any non-secret base element has the universal set as its secrecy group.

3.6.3 Schemes for Basic Properties and Proofs

The previous section presented the basic security properties and the mechanisms for enforcing them. This section summarizes the schemes and the mechanisms used for various properties in Table 3.10, before proving them using SPCL. Proofs based on primitive actions are presented in the next section.

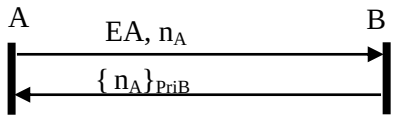
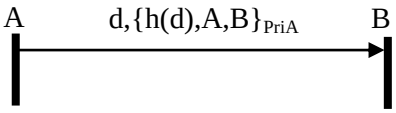
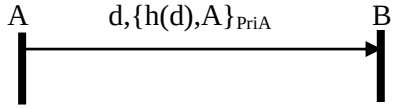
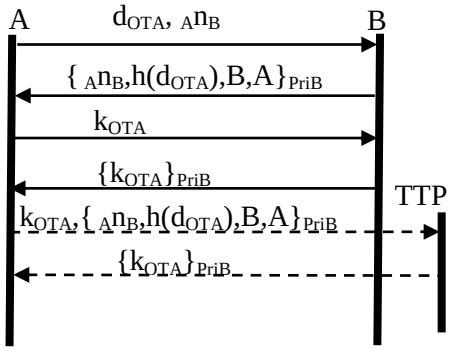
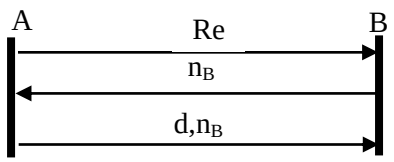
Property	**Scheme**	**Mechanism**
Entity Authentication		If *A* receives its recent nonce encrypted by the private key of *B* then *A* can be certain about the peer's identity (i.e., it is *B*).
Data Authentication		When *B* receives data together with a digest containing a hash of that data and labels *A* and *B*, encrypted by *A*'s private key, *B* can be certain that it had originated at *A* and that it was directed to *B*.
Data Integrity		If *B* receives data accompanied with a digest containing a hash of that data and label *A*, encrypted by *A*'s private key, *B* can be certain that the data from *A* has not been tampered with.
Receiver Non-Repudiation		If *A* receives two messages, the first a digest made up of an earlier nonce, the hash of data-encrypted-with-one-time-key (d_{OTA}) and labels *B* and *A*, and the second with the one-time key, both encrypted by *B*'s private key, then *B* cannot repudiate receiving the message. The nonce makes the protocol run unique. Deferring the despatch of k_{OTA} until after the receipt of acknowledgement makes the exchange fair. The last two steps involving a trusted third party is carried out only if no acknowledgement is received from B. The evidence presented to TTP and the key k_{OTA} allows it to extract receipt of acknowledgement from *B* (through TTP).
Recency		If *B* receives data together with the digest containing the nonce that it had generated recently, then that data must be recent.

Table 3.10 Summary of Schemes/Mechanisms for Basic Properties

3.6.3.1 Proof System

This section uses SPCL to reason about protocol actions. SPCL allows assertions to be made based on a series of protocol actions which include primitive actions such as *send*, *receive*, *new*, *encrypt*, *decrypt*. These assertions provide security guarantees to both data originator and recipient and can be verified by using the proof system developed by Hoare based on preconditions and invariants [77] (presented in Section 2.3.2.2). The same proof system is also used to verify the composite schemes formed for multiple properties in Section 3.6.4. In addition to the action formulas, the formula *Knows(X,T)* and *Assert(p)* are used in this model. The formula *Knows(X,T)* states *X* knows the term *T*, either because it originated in *X* or was made known through data receipt or decryption. The formula *Assert(p)* states the predicate *p* can be proved using available evidence. All the proofs in this section make the **perfect encryption assumption** defined in Section 2.1.2. Moreover, **honest entities** are assumed to adhere to protocol specifications and assumptions.

DEFINITION 3.2: PGPS PRIMITIVE ACTIONS

PGPS actions are expressed in terms of *Send*, *Rec*, *Enc*, *Dec* and *New* as in:

A ::= Send(A,B,M)|Rec(A,B,M)|New(A,N)|New(A,OTK)|Enc(A,T,PriA)|Dec(A,T,PubTO)

where the primitive actions *Send(A,B,M)* and *Rec(A,B,M)* represent message *M* sent from *A* and received by *B*, and message *M* received by *A* sent from *B*, respectively.

- The primitive actions *New(A,N)* and *New(A,OTK)* allows the creation of a new nonce *N* or one-time-key *OTK* in entity *A*, respectively.
- The primitive action *Enc(A,T,priA)* represents encryption of term *T* by *A*'s private key.
- The primitive action *Dec(A,T,pubTO)* represents *A*'s decryption of term *T* by the public key where *T* originates.

DEFINITION 3.3: PGPS ASSERTIONS

In PGPS *Assert(p)* asserts predicate *p* is true where *p* can be constructed from predicates *Origin*, *Dest*, *DataValid* and *Precede* as in:

p::=*Origin(T,TO)*|*Dest(T,TR)*|*DataValid(T,TO)*| *Precede(M1,M2)*

where:

- *Origin(T,TO)* and *Dest(T,TR)* assert *T*'s origin (*TO*) and intended destination (*TR*) respectively.
- *DataValid(T,TO)* asserts integrity of term *T* received from origin *TO*.
- *Precede(M1,M2)* asserts message *M1* causally precedes message *M2* in the same entity.

DEFINITION 3.4 PGPS INFERENCE RULES

PGPS inference rules are expressed in the form of axioms involving primitive actions and assertions as shown in Table 3.11. The notations used in these axioms are described below.

PA1	$New(X,N1) \wedge New(Y,N2) \supset N1 \neq N2$ Nonces created by two different entities are never identical
PA2	$New(X,N) \supset Knows(X,N)$ New information generated is added to the entity knowledge
PA3	$Rec(X,Y, M \leftarrow [T1, T2, \ldots Tn]) \supset Knows(X, T_1) \wedge Knows(X, T_2) \ldots Knows(X, Tn)$ Information received is added to the entity knowledge
PA4	$Knows(X,T1) \wedge Knows(X,T2) \supset Knows(X,[T1,T2])$ Knowledge can be compounded
PA5	$Knows(X,[M1,M2]) \supset Knows(X,M1) \wedge Knows(X,M2)$ Knowledge can be decomposed
PA6	$Send(X,Y,M = [T1, T2, \ldots Tn]) \supset Knows(X,T_1) \wedge Knows(X,T_2) \ldots Knows(X,Tn)$ Only known terms can be sent
PA7	$Rec(R,Y,[EMD,d]) \wedge MD \leftarrow Dec(R,EMD,pubY) \supset Assert(Enc(Z,MD, priY))$ When *R* decrypts the *EMD* from *Y* using its public key and form *MD*, *R* can assert some entity *Z* has encrypted *MD* with the private key of *Y*
PA8	$Assert(Enc(Z,MD, priY)) \wedge Honest(Y) \supset Assert(Z = Y)$ Asserting that entity *Z* encrypted the digest *MD* with the private key of *Y*, and *Y* is honest (Y will not divulge its private key) allows *Z = Y* to be asserted
PA9	$Assert(Enc(Y,MD, priY)) \wedge MD.dest = R \supset Assert(Dest(M,R))$ The intended destination of message *M* can be asserted based on the label in the signed digest
PA10	$Assert(Enc(Y,MD, priY)) \wedge MD.source = Y \supset Assert(Origin(M, Y))$ The origin of message *M* can be asserted based on the origin label in the signed digest
PA11	$Assert(Origin(M,X)) \wedge Mi \in M \supset Assert(Origin(Mi,X))$ The origin of message component (part) *Mi* can be asserted based on origin of message *M*
PA12	$Assert(Enc(Y,MD,priY)) \wedge MD.source=Y \wedge MD.hash=h(d) \supset Assert(DataValid(M,Y))$ The validity of data can be asserted based on validity of hash of data received
PA13	$TimeReceived(X,M1) < TimeSent(X,M2) \supset Assert(Precede(M1,M2))$ The ordering of two messages can be asserted if the second message is not despatched until after the receipt of first message (regardless of the type of network)
PA14	$Rec(R,Y,[N,M]) \wedge new(R,N) \supset Assert(Recent(M))$ The recency of message can be asserted based on the recency of the nonce

Table 3.11 Protocol Axioms

$M \equiv$ **M**essage – consists of one or more terms made up of data and digests

$MD \equiv$ **M**essage **D**igest – combines some of the attributes below depending on the scheme used.

- ***hash*** – hash of data
- ***nonce*** – unique value generated for binding or for proving recency
- ***origin***, ***dest*** – labels for data origin and destination

$EMD \equiv$ **E**ncrypted **M**essage **D**igest – formed by signing MD by private key of message originator

The operators . (dot), $=$, $\leftarrow$ and $[]$ are used for attribute, equality, assignment and concatenation respectively. For example:

- $y = MD.p1$ returns true if y *is* equal to attribute $p1$ of MD
- $MD \leftarrow Dec(X,EMD,key)$ means X decrypts EMD with key and assigns it to MD
- $M \leftarrow [T1,T2]$ means M is formed concatenating $T2$ to $T1$

Assumptions and Invariants

The assumptions and invariants used by security schemes are listed in Table 3.12. Invariants restrict the data elements that can be learnt by the intruder and valid participants. Invariants for basic schemes can vary reflecting the properties they enforce. Basic schemes can be combined safely to provide multiple properties only if the individual schemes do not violate the invariants of one another [26].

Assumptions and Invariants	Description
A1	All recipients posses the public key of originators
A2	All senders posses the public key of recipients (needed for receiver non-repudiation)
A3	Explicit labels for message originator and recipient in the part encrypted by private key can prevent the man in the middle (MITM) attack, where an adversary responds with a message faking its identity.
Inv1	Private and group keys are not sent in the open
Inv2	All nonces (random numbers) despatched are freshly generated
Inv3	Data must not be released completely until acknowledgement received for partial release
Inv4	One-time keys used in non-repudiation are not reused for encryption
HEA (Honest Entity Assumption)	All honest entities adhere to the invariants Inv1, Inv2, Inv3, Inv4

Table 3.12 Assumptions and Invariants

This section presents proofs for various two-party schemes which allow the protocol initiator and responder to assert specific security properties. For each of the basic security properties, the series of primitive actions used in the scheme is presented sequentially. A combination of public keys, symmetric keys, hashes and nonces are used to enforce the security properties. Note that distinct labels are used for message digests (MD_1, MD_2, …) and encrypted message digests (EMD_1, EMD_2, …) as distinct schemes use different elements in the digests. These proofs require showing that if preconditions and invariants are true at the beginning after the protocol actions post-conditions and invariants will remain true as in: {Preconditions $\wedge$ Invariants} Protocol-Actions {Post-conditions $\wedge$ Invariants}.

1. Entity Authentication (EA) Scheme (*A*'s actions carried out as part of Authenticating *B*)

Precondition: ***Honest(A,B),*** **Invariants:** ***Inv1, Inv2***

L1. $New(A, n_A)$ // *A* creates new nonce

L2. $Send(A,B, ["EA", n_A])$ // *A* sends the string "*EA*" and fresh nonce n_A

L3. $Receive(A,B, EMD_1)$ // *A* receives the encrypted message digest EMD_1 containing n_A

L4. $MD_1 \leftarrow Dec(A,EMD_1,pub_B)$ // *A* extracts the message digest (n_A) by decrypting EMD_1 with pub_B

L5. $MD_1.nonce = n_A$ // *A* Verifies that the nonce received ($MD_1.nonce$) is the same as n_A

Post-condition: ***Honest(A,B); A Assert(entity authentication by B)***

Invariants: ***Inv1, Inv2***

Proof: The Entity Authentication Schemes allow *A* to Authenticate *B*

By precondition ***Honest(A,B)*** and honest entity assumption **HEA,** invariants **Inv1** and **Inv2** are initially true. By axiom PA7, MD_1 in **L4** derived by decrypting message *EMD1* using the public key of *B*, was despatched by an entity in possession of private key of *B*. Based on the precondition *B* is honest and by assumption **HEA** (Table 3.12) Inv1 holds (private keys are kept confidential), it follows *MD1* originated in *B*. Furthermore, by axiom **PA1** the newly generated nonce n_A is unique. Receipt of n_A as a part of the term encrypted by the private key of *B* (in **L4** and **L5**), allows *A* to assert that the peer has successfully responded to the nonce challenge, and has authenticated itself to be *B*. The invariant **Inv1** is not violated by *A*, as the only message sent out in **L2** contains only a string constant ("EA") and a nonce. The invariants **Inv1** is not violated by *B* as the only message received from *B* in **L3**, is not sent in the open. The invariant **Inv2** is not violated by *A* as the nonce sent out by *A* in **L2 is** freshly generated in **L1.** The invariant **Inv2** is not violated by *B* as it does not despatch any nonce. □

2. Data Authentication (A) Scheme (*B*'s actions when receiving *A*'s authenticated data)

Pre-condition: ***Honest(A,B)*** **Invariants:** ***Inv1***

L1. *Receive(B,A,[d,*EMD_2*])* // *B* receives data *d* and encrypted message digest EMD_2 *from A*

L2. MD_2 ←*Dec(B,*EMD_2*,*pub_A*)* // *B* decrypts EMD_2 with pub_A and stores in MD_2

L3. MD_2*.hash=h(d)* MD_2*.origin=A* MD_2*.dest=B* // verifying message hash, origin and destination

Post-condition: ***Honest(A,B); B Assert(Data Authentication by A)***

Invariants: ***Inv1***

Proof: The Authentication Schemes allows *B* to Authenticate *A*'s Data

By precondition ***Honest(A,B)*** and honest entity assumption **HEA,** invariant **Inv1** is initially true. By axiom PA7, MD_2 in **L2** derived by decrypting message *EMD2* using the public key of *A*, was despatched by an entity in possession of private key of *A*. Based on the precondition *A* is honest and by assumption **HEA** (Table 3.12) **Inv1** holds, (private keys are kept confidential), it follows *MD2* originated in *A*. Axioms **PA9**, **PA10** and **PA12** allow *B* to assert the validity of the intended recipient, the origin and data, based on the receipt of labels for recipient and origin, and hashed data value in the decrypted part (in **L3)**, allowing *B* to assert data authentication of message from *A*. Invariant **Inv1** is not violated by *A* as it does not send any message as part of protocol action. Invariant **Inv1** is not violated by *B* as it does not send any data in the open. □

3. Data Integrity (DI) Scheme (*B*'s actions when receiving data from *A* with data integrity)

Pre-condition: ***Honest(A,B)*** **Invariants:** ***Inv1***

L1. *Receive(B,A,[d,*EMD_3*])* // receiving the data d and signed message digest EMD_3 *from A*

L2. MD_3 ←*Dec(B,* EMD_3*,*pub_A*)* // *B* decrypts EMD_3 with pub_A and stores in MD_3

L3. MD_3*.hash=h(d)* MD_3*.sender=A* // verifying message hash and origin

Post-condition: ***Honest(A,B); B Assert(Integrity of Message received from A)***

Invariants: ***Inv1***

Proof: The Data Integrity Scheme allows *B* to Verify Integrity of *A*'s Data

By precondition ***Honest(A,B)*** and honest entity assumption **HEA,** invariant **Inv1** is initially true. By axiom PA7, MD_2 in **L2** derived by decrypting message $EMD2$ using the public key of A, was despatched by an entity in possession of private key of A. Based on the precondition A is honest and by assumption **HEA** (Table 3.12) **Inv1** holds (private keys are kept confidential), it follows $MD2$ originated in A. Axioms **PA10** and **PA12** allow B to assert the origin and integrity of data, based on the receipt of the label for origin, and the hashed data value in the decrypted part (in **L3)**, allowing B to assert data authentication of message from A. Invariant **Inv1** is not violated by A as it does not send any message as part of protocol action. Invariant **Inv1** is not violated by B as it does not send any data in the open. □

4. Receiver Non-Repudiation (RNR) Scheme (*A*'s actions to non-repudiate *B*'s receipt of its data)

Pre-condition: ***Honest(A,B)***	**Invariants:** ***Inv1,Inv2,Inv3,Inv4***
L1. $New(A,{}_An_B)$	// A creates a nonce to ensure unique run with B
L2. $New(A,k_{OTA})$	// A creates a new one-time key for encrypting data
L3 $d_{OTA} \leftarrow Enc(A,d,k_{OTA})$	// A encrypts data with one-time key
L4. $Send(A,B,[d_{OTA},{}_An_B])$	// A sends both encrypted data (by one-time key) and nonce to B
L5. $Receive(A,B,\ EMD_4)$	// receive encrypted digest containing data hash, nonce and labels
L6. $MD_4 \leftarrow Dec(A,EMD_4,pub_B)$	// A decrypts encrypted message digest storing it in MD_4
L7. $MD_4.hash=h(d_{OTA})MD_4.nonce={}_An_B\ MD_4.sender=B\ MD_4.receiver=A$	//verify hash, nonce & labels
L8. $Send(A,B,k_{OTA})$	// sending the one-time-key to recipient (assuming previous step valid)
L9. $Receive(A,B,EMD_{4B})$	// receiving the second encrypted message digest
L10. $MD_{4B} \leftarrow Dec(A,EMD_{4B},pub_B)$	// *A* decrypts encrypted message digest storing it in MD_{4B}
L11. $MD_{4B}.key=\ k_{OTA}$	// verifies receipt of key by B. L7 and L11 ensures B cannot repudiate
---------- **Alternative to L9 – L11 (using TTP)** ----------	
L12. $Send(A,TTP,[k_{OTA},EMD_4])$	// sending the one-time-key to recipient (assuming previous step valid)
L13. $Receive(A,TTP,EMD_{4C})$	// receiving the second encrypted message digest indirectly through TTP
L14. $MD_{4C} \leftarrow Dec(A,EMD_{4C},pub_B)$	// *A* decrypts encrypted message digest storing it in MD_{4C}
L15. $MD_{4C}.key=\ k_{OTA}$	// Verifies receipt of key by B. L7 and L14 ensures B cannot repudiate
---------- ----------	
Post-condition: ***Honest(A,B); A Assert(Message Non-Repudiation by B)***	
Invariants: ***Inv1,Inv2,Inv3,Inv4***	

Proof: Non-repudiation Scheme allows *A* to Non-Repudiate *B*'s receipt of its Data

By precondition ***Honest(A,B)*** and honest entity assumption **HEA,** invariants **Inv1**, **Inv2**, **Inv3** and **Inv4** are initially true. The proof proceeds in two steps as data is partially released with encrypted data despatched in the initial stage and the key in the second stage. By axiom PA7, MD_4 in **L6** derived by decrypting message $EMD4$ using the public key of B, could only be despatched by an entity in possession of private key of B. Since the precondition assumes B is honest and that by assumption

HEA (Table 3.12) honest entities keep their private keys confidential, it follows $MD4$ originated in B. Axioms **PA9** and **PA10** allow the intended recipient and originator of B's responses MD_{4A} and MD_{4B} to be associated with A and B respectively. The axiom **PA12** allows receipt of encrypted data (**L7**) to be proved based on the hash of the data in the encrypted part of the first response while axiom **PA11** makes the receipt of key by B undeniable based on the second response (**L11**). The receipt of nonce ${}_An_B$ and the axiom **PA1** asserting uniqueness of a nonce, allow the response to be associated with a specific run. Thus, A can assert non-repudiation by B. If A does receive the second encrypted message digest in step 9, the recovery phase in steps **L12** to **L15** are initiated, where the first encrypted message digest forming the evidence and the key k_{OTA} are passed to the TTP. The TTP uses the evidence and the key to extract non-repudiable evidence from recipient B [7] before forwarding it back to A. By axiom **PA11,** the key part of response MD_{4C} (in lines **L14** and **L15**) received indirectly through **TTP** could only have originated in B**,** making it impossible for B to deny receipt of data. The invariant **Inv1** is not violated as the only key sent in the open by either A or B *is a* one-time key. The invariant **Inv2** is not violated as the only nonce, ${}_An_B$ is freshly generated by A in **L1**. Similarly **Inv3** is not violated as the key is not released until after the receipt and verification of acknowledgement in lines **L5** to **L7**. **Inv4** is not violated as the only one-time key sent (k_{OTA}) is freshly generated in **L2**. □

5. Recency (R) Scheme (*B*'s actions when receiving recent data)

Pre-condition: *Honest(A,B)* **Invariants:** ***Inv1, Inv2***	
L1. `Receive (B, A,"Re")`	// B receives request to send data with recency assurance from A
L2. `New (B, n_B)`	// B creates fresh nonce to enforce recency
L3. `Send(B,A, n_B)`	// B responds to A's request by sending the nonce
L4. `Receive(B,A,[d, n_B])`	// receipt of data together with recently generated nonce sent earlier
Post-condition: ***Honest(A,B);*** `B Assert (Data Recency)` (not necessarily from A)	
Invariants: ***Inv1, Inv2***	

Proof: B can verify Recency of Data

By precondition ***Honest****(A,B)* and honest entity assumption **HEA,** invariants **Inv1** and **Inv2** are initially true. By axiom **PA1,** nonce n_B generated in B (in **L2**) is unique. By axiom **PA14**, despatch of a message by A incorporating a nonce recently generated and sent, must be recent. Invariant **Inv1** is not violated by A as the messages received from A in **L1** and **L4** do not contain any keys. Invariant **Inv1** is not violated by B as the only message sent by B in **L3** contains only the nonce n_B. Invariant **Inv2** is not violated by either A or B as the only nonce, n_B is freshly generated by *B* in **L2.**

Note the recency scheme, however, does not guarantee the message originated from A. In the following section the recency scheme is combined with other basic schemes to prevent such attacks.

3.6.4 Composite Properties and Schemes Enforcing Them

The security level for each message in PGPS is set by aggregating the security requirements of the data originator and recipients. Hence, a security level may contain two or more security properties. If such multiple properties are to be enforced using composite properties, their meaning must be made precise to avoid misinterpretations. Different levels of assurances for common properties led to protocol flaws in the past through misinterpretation [13]. PGPS uses emergent behaviour (Terminology 3.1) to standardize the order in which each basic property must be enforced. Furthermore, the cryptographic schemes devised for them are made to enforce emergent behaviours. Section 3.6.4.1 presents the emergent behaviours used in PGPS. Section 3.6.4.2 outlines the techniques for composing non-interfering schemes either in parallel or in sequence. Section 3.6.4.3 presents techniques devised for reducing computational costs and bandwidth. Section 3.6.4.4 illustrates the process used for creating composite schemes. Section 3.6.4.5 lists the schemes created using these techniques.

The main steps in creating composite security schemes are outlined below.

1. Identify the invariants and assumptions upon which schemes for individual properties are based. Section 3.6.3.1 gave proofs that security goals are met when these invariants and assumptions hold for the underlying schemes. For example, the data integrity scheme was proved assuming invariant Inv1 and assumption HEA hold (refer to Table 3.12).
2. Combine non-conflicting schemes either in parallel or in series to create schemes providing multiple security assurances. The scheme for entity authentication is composed with schemes for all other properties in sequence, as the identity of the responder must be verified before despatching any data (precondition). All other schemes are combined in parallel after ensuring that individual schemes do not interfere with the invariants of one another.
3. If basic schemes are in conflict, then non-interfering schemes must be devised. In 3.6.4.4 extended schemes are devised for authentication, integrity and recency before combining them with the scheme for non-repudiation.
4. When the same type of cryptographic elements (such as nonces and keys) are used in basic schemes, distinct elements (such as two different nonces) must be created to ensure non-interference in combined schemes [76].

3.6.4.1 Standardised Emergent Behaviours

Basic security properties in PGPS are classified into those providing assurances to the data originator (receiver non-repudiation, entity authentication) and data recipients (authentication, data integrity, recency). Emergent behaviours (Terminology 3.1) for composite properties are standardised based on their usefulness for e-commerce. For example, the standardised emergent behaviour for the composite property combining authentication and receiver non-repudiation requires that data authentication be asserted first, because non-repudiable of evidence of receipt should not be sent until authenticity of the data can be verified. Furthermore, by ensuring that all parties have a common, consistent interpretation

of composite security properties, PGPS makes reasoning about security needs for messages possible. Table 3.13 shows the order in which individual security properties are enforced in PGPS. Entity authentication assurance is enforced first, as the identity of peer entity must be verified before any data are despatched. Similarly, authentication, integrity and recency properties are enforced before receiver non-repudiation, as the receiver may want these assurances before providing non-repudiable evidence of data receipt. Hence, any property combined with non-repudiation provides a much stronger assurance for the data originator, than non-repudiation alone. For example, when recency is combined with non-repudiation, message originator gets the assurance that data was received and that it was received in a timely manner.

	First	Second	Third
Entity Authentication	X		
Authentication		X	
Data Integrity		X	
Recency		X	
Non-Repudiation			X

Table 3.13 Emergent Behaviours Specifying Order of Enforcement for Security Properties

3.6.4.2 Techniques for Enforcing Non-Interference

Schemes for individual security properties can be combined to provide multiple properties only when they do not interfere with each other [25]. Interference with secrecy is avoided in PGPS by using independent schemes at data element level, before applying the schemes for other properties. Extended schemes are devised for correspondence properties in PGPS if the basic schemes to be combined violate the invariants of one another. The basis for validity of such an approach was presented in Chapter 2 and is summarised below:

- Schemes for multiple security properties can be created by combining basic schemes in parallel as long as the basic schemes do not violate the invariants of each other.
- Schemes can be combined in sequence, if in addition the post-condition of the first scheme meets the precondition of the second scheme.

3.6.4.3 Reducing Overheads in Combined Schemes

When security schemes are combined directly to provide multiple assurances, additional redundant elements are usually introduced. However, past attempts to optimise such schemes by removing duplicate elements within encrypted parts has resulted in introducing new avenues for attacks [21]. Therefore, the impact of these changes on security goals must be carefully analysed before removing or combining redundant elements. The guidelines used in PGPS when combining schemes are outlined below:

- Repeated labels and cryptographic components in plaintext can be removed as they provide no additional information. However, repeated items in distinct encrypted parts are not removed as they may provide non-repudiable evidence associating data with entities.

- Encrypted terms that are identical can be removed as they carry no additional guarantees.
- Data elements contained in distinct terms signed by the private key of the originator can be combined when there is only one data recipient. Such elements should not be combined if the data is intended to reach multiple recipients in sequence, as the evidence placed in different terms may be intended for different recipients.

3.6.4.4 Forming Composite Schemes

Most basic schemes presented in Section 3.6.3 avoid interference with each other's invariants. The non-repudiation scheme, however, has specific invariants related to fair exchange requirements which are violated by other schemes. Table 3.14 shows all the properties and invariants violated by those schemes, and the reason why the invariant is violated. For example, the scheme used for authentication conflicts with that of non-repudiation, which prevents full release of data until partial acknowledgement is received. In the refined schemes presented in the next section for authentication, data integrity and recency, data is released in parts, thus avoiding conflicts with non-repudiation invariants.

Scheme	Violated Invariants	Reason
Authentication (basic)	Inv3	Authentication scheme does not use partial release
Data Integrity (basic)	Inv3	Data integrity scheme does not use partial release
Recency (basic)	Inv3	Recency scheme does not use partial release
Non-Repudiation	None	
Secrecy	None	
Entity Authentication	None	

Table 3.14 Invariants Violated by Various Security Properties

Revised Schemes for Authentication, Data Integrity and Recency

The revised schemes extended to allow partial release of data for authentication, data integrity and recency are shown in Figure 3.10. These extended schemes are used in composition only when the basic schemes violate the invariants of others, as the additional steps involved make them costlier. Partial release is facilitated by sending data encrypted with a one-time key in the first pass, followed by the key after the receipt of acknowledgement. These refined schemes can be combined with any of the other schemes, as they do not violate any of the invariants identified earlier. For example, none of the three schemes below violate invariant *Inv3*, which prevents the release of a one time-key until the recipient's acknowledgement of encrypted data.

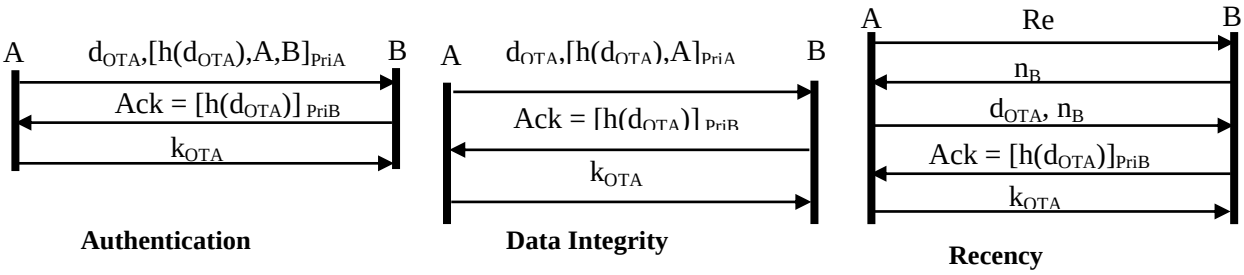


Figure 3.10 Refined Schemes

Composition Rules

The vertical axis in Figure 3.11 shows the five message-response cycles (2 steps each) along which schemes for entity authentication, recency, authentication, data integrity and non-repudiation are merged, to enforce the emergent behaviours specified in Table 3.13. The black nodes show the points at which the property is first asserted. For example, authentication, data integrity and recency properties are asserted before non-repudiation. Note the last cycle for non-repudiation involving TTP is necessary only when no acknowledgement for the key is received (refer to Figure 3.9). Data integrity scheme is shown together with data authentication scheme as they have the same sequence (refer to Sections 3.6.2.1 and 3.6.2.3). Extended schemes (shown for recency, data-integrity and authentication) are used when the composite property includes non-repudiation, to ensure the invariants for individual schemes are not in conflict. Figure 3.11 also shows that the entity-authentication scheme must be completed before any of the other schemes are implemented as its post-condition meets the precondition for others (i.e., the validity of the responder is verified before sending any data). Therefore the entity authentication scheme is merged sequentially with the others, while all other schemes are merged in parallel as their schemes are overlapping.

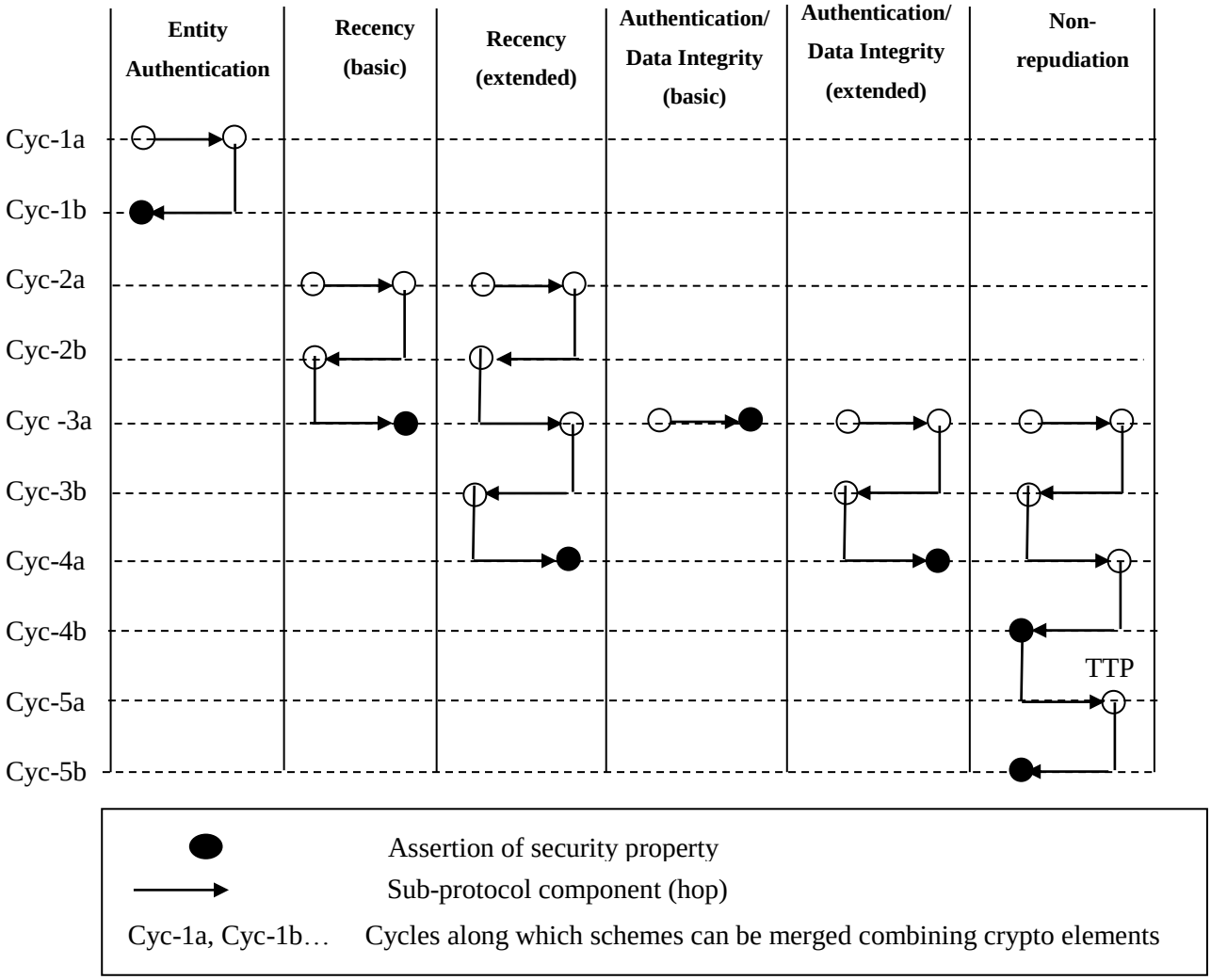


Figure 3.11 Merging Security Schemes

3.6.4.5 Composed Schemes

Tables 3.15 to 3.25 show the schemes for fine-grained security properties derived from composition techniques described in Section 3.6.4.4. These schemes also meet the emergent behaviours specified in Table 3.13. The schemes for entity authentication property composed with others are not shown, as it is the only scheme in the first cycle which must be completed before any other scheme, as shown in Figure 3.11. The validity of combined schemes follows directly from the validity of schemes used for individual properties as long as their invariants are not violated.

Composed Scheme: Data Authentication and Data Integrity

This composed scheme shown in Table 3.15, assures the recipient of the authenticity and integrity of data. It is identical to the scheme used for authentication, as the guidelines for reducing overheads in merged schemes presented in 3.6.4.3, permit combination of encrypted parts and removal of repeated elements.

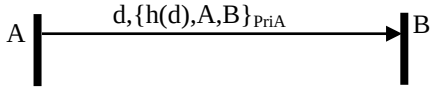
Proof basis	Composed Scheme	Mechanism
A1, A3, Inv1, HEA		If the hash part in the digest signed by message originator matches the hash of data received, *B* can be assured of its integrity. In addition, if the encrypted part also contains the labels for message originator *A* and recipient *B*, *A* cannot deny authenticating the data to *B*.

Table 3.15 Composed Scheme for Authentication and Integrity

Composed Scheme: Data Authentication and Receiver Non-Repudiation

This composed scheme shown in Table 3.16, assures the recipient authenticity of the data and the originator non-denial of data receipt. Note the use of TTP is necessary only when *A* does not receive the acknowledgement for the key k_{OTA} from *B*.

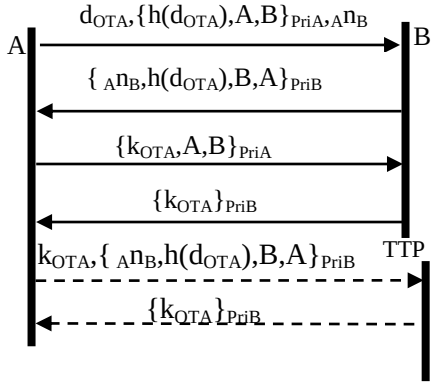
Proof basis	Composed Scheme	Mechanism
A1, A2, A3, Inv1, Inv2, Inv3, Inv4, HEA		If *A* receives from *B* two signed messages one consisting of a hash of data encrypted with a one-time-key ($h(d_{OTA})$) and nonce sent earlier (ensures a unique run), and the other, the key needed for decryption, then *B* cannot repudiate receiving the data. *A* cannot deny authenticating the (encrypted) data to *B*, as the digest signed by *A* consists of hash of (encrypted) data and labels for *A* and *B*. Although step 1 sends the nonce $_An_B$ in the open, the use of explicit labels for responder and initiator in step 2 prevents MITM attacks (A3)

Table 3.16 Composed Scheme for Authentication and Receiver Non-Repudiation

Composed Scheme: Data Authentication and Recency

This composed scheme in Table 3.17, assures the recipient authenticity and recency of data. The term *Re* in the first step requests a nonce challenge for proving recency.

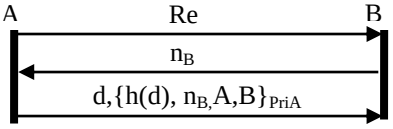
Proof basis	Composed Scheme	Mechanism
A1, A3, Inv1, Inv2, HEA		A digest signed by the private key of *A* containing a hash of the data and *B*'s recent nonce allows *B* to prove *A* has authenticated the data recently. Although step 2 sends the nonce n_B in the open, MITM attacks are prevented through explicit labels in the encrypted part in step 3 (A3)

Table 3.17 Composed Scheme for Authentication and Recency

Composed Scheme: Data Integrity and Receiver Non-Repudiation

This composed scheme in Table 3.18, assures the recipient of the integrity of the data and the originator non-denial of data receipt. Note the use of TTP is necessary only when *A* does not receive the acknowledgement for the key k_{OTA} from *B*.

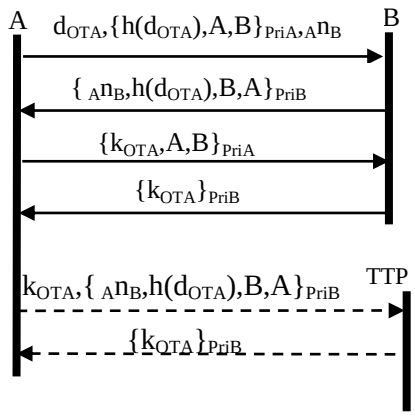
Proof basis	Composed Scheme	Mechanism
A1, A2, A3, Inv1, Inv2, Inv3, Inv4, HEA		*B* cannot repudiate receiving the data as *A* receives two signed messages from *B*: the first made up of a hash of encrypted data ($h(d_{OTA})$) together with a nonces ensuring unique run and recency, and the second with the key needed for decryption. If the hash part in the digest signed by the message originator *A* matches the hash of data received, *B* can be assured of its integrity. Use of explicit labels in steps 1 to 3 prevents MITM attacks (A3).

Table 3.18 Composed Scheme for Integrity and Receiver Non-Repudiation

Composed Scheme: Data Integrity and Recency

This composed scheme in Table 3.19, assures the recipient integrity and recency of data.

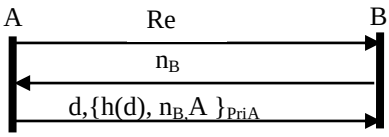
Proof basis	Composed Scheme	Mechanism
A1, A3, Inv1, Inv2, HEA		A digest signed by the private key of *A* containing a hash of the data and a recent nonce from *B* sent earlier, allows *B* to prove *A* has sent the data recently. If the hash part in the digest signed by the message originator matches the hash of data received, *B* can be assured of its integrity.

Table 3.19 Composed Scheme for Integrity and Recency

Composed Scheme: Recency and Receiver Non-Repudiation

This composed scheme in Table 3.20, assures the recipient of the recency of the data and the originator non-denial of data receipt. As the recipient verifies data recency before acknowledgement, the data originator can use that acknowledgement as evidence of data receipt with recency. Note the use of TTP is necessary only when *A* does not receive the acknowledgement for the key k_{OTA} from *B*.

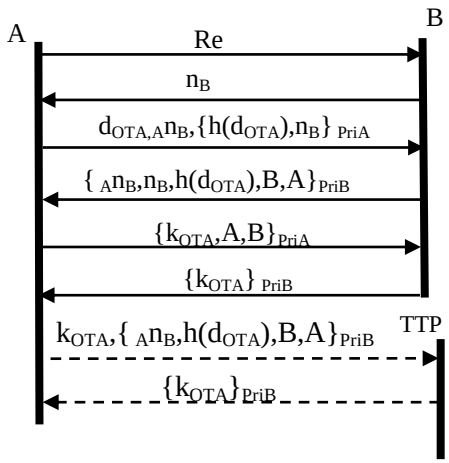
Proof basis	Composed Scheme	Mechanism
A1, A2, A3, Inv1, Inv2, Inv3, Inv4, HEA		*B* can prove the data received is recent if it is accompanied by a signed digest containing *B*'s nonce (sent earlier) with the hash of the data. If *A* receives from *B* signed messages consisting of a hash of encrypted data together with nonces for ensuring unique run and recency, and the key needed for decryption, then *B* cannot repudiate receiving the data. Although steps 2 and 3 send nonces in the open, MITM attacks are prevented through explicit labels in the encrypted part in steps 3 and 4 (A3)

Table 3.20 Composed Scheme for Recency and Receiver Non-Repudiation

Composed Scheme: Data Authentication, Data Integrity and Receiver Non-Repudiation

This composed scheme in Table 3.21, assures the recipient of the integrity and authenticity of the data and the originator non-denial of data receipt. Note the use of TTP is necessary only when *A* does not receive the acknowledgement for the key k_{OTA} from *B*.

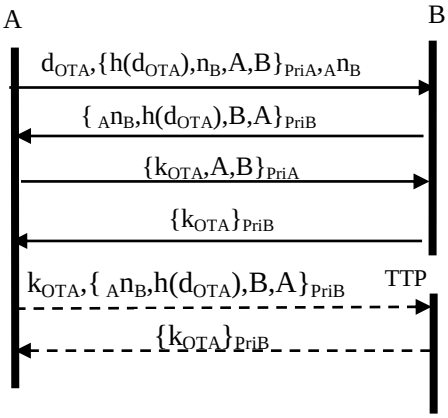
Proof basis	Composed Scheme	Mechanism
A1, A2,A3 Inv1, Inv2, Inv3, Inv4, HEA		*B* can prove the integrity of the data and its authenticity as it is accompanied by a signed digest containing the hash of the data together with labels for message originator *A* and intended recipient *B*. *B* cannot repudiate receiving the data as *A* receives from *B* two signed messages: the first made up of a hash of the encrypted data together with a nonce ensuring a unique run, and the second with the key needed for decryption. Use of explicit labels in step 2 prevents MITM attacks (A3).

Table 3.21 Composed Scheme for Authentication, Integrity and Receiver Non-Repudiation

Composed Scheme: Data Authentication, Receiver Non-Repudiation and Recency

This composed scheme in Table 3.22, assures the recipient of the recency and authenticity of the data and the originator non-denial of data receipt. Note the use of TTP is necessary only when *A* does not receive the acknowledgement for the key k_{OTA} from *B*.

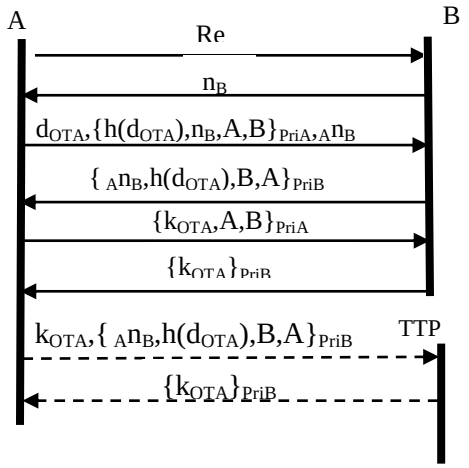
Proof basis	Composed Scheme	Mechanism
A1, A2,A3 Inv1, Inv2, Inv3, Inv4, HEA		*B* can prove that the data received is recent and authenticated as it is accompanied by a signed digest containing the hash of (encrypted) data together with *B*'s nonce (sent earlier) and labels for the message originator *A* and the intended recipient *B*. *B* cannot repudiate receiving the data as *A* receives from *B* two signed messages: the first made up of hash of encrypted data together with nonces ensuring a unique run and recency, and the second with the key needed for decryption. Use of explicit labels in steps 3 and 4 prevents MITM attacks (A3).

Table 3.22 Composed Scheme for Authentication, Receiver Non-Repudiation and Recency

Composed Scheme: Data Authentication, Data Integrity and Recency

This composed scheme in Table 3.23, assures the recipient integrity, authenticity and recency of data received.

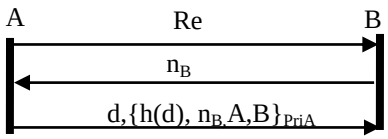
Proof basis	Composed Scheme	Mechanism
A1, A3 Inv1, Inv2, HEA		*B* can prove integrity, recency and authenticity of the data received as it is accompanied by a signed digest containing the hash of data together with *B*'s nonce (sent earlier) and labels for the message originator *A* and the intended recipient *B*. Use of explicit labels in step 3 prevents MITM attacks (A3).

Table 3.23 Composed Scheme for Authentication, Integrity and Recency

Composed Scheme: Data Integrity, Receiver Non-Repudiation and Recency

This composed scheme in Table 3.24, assures the recipient of the recency and integrity of the data and the originator non-denial of data receipt. The composed scheme is shown in Table 3.24. Note the use of TTP is necessary only when *A* does not receive the acknowledgement for the key k_{OTA} from *B*.

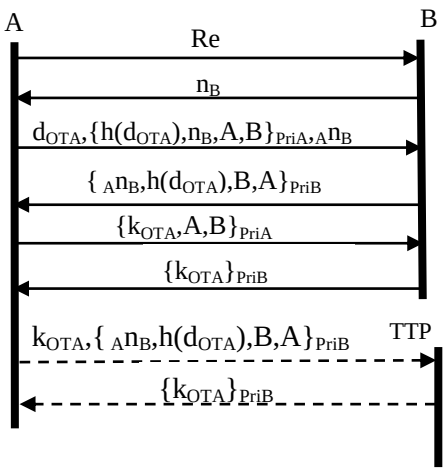
Proof basis	Composed Scheme	Mechanism
A1, A2,A3 Inv1, Inv2, Inv3, Inv4, HEA		*B* can prove integrity and recency of data as it is accompanied by a signed digest containing the hash of data together with *B*'s nonce (sent earlier) and label for message originator *A*. *B* cannot repudiate receiving the data as *A* receives from *B* two signed messages: first made up of hash of encrypted data together with a nonces ensuring unique run and recency, and the second with key needed for decryption. Use of explicit labels in step 3 and 4 prevent MITM attacks (A3).

Table 3.24 Composed Scheme for Integrity, Receiver Non-Repudiation and Recency

Composed Scheme: Data Authentication, Data Integrity, Receiver Non-Repudiation and Recency

This composed scheme in Table 3.25, assures the recipient of the recency, authenticity and integrity of the data, and assures the originator that data receipt cannot be denied. The composed scheme is shown in Table 3.25. Note the use of TTP is necessary only when *A* does not receive the acknowledgement for the key k_{OTA} from *B*.

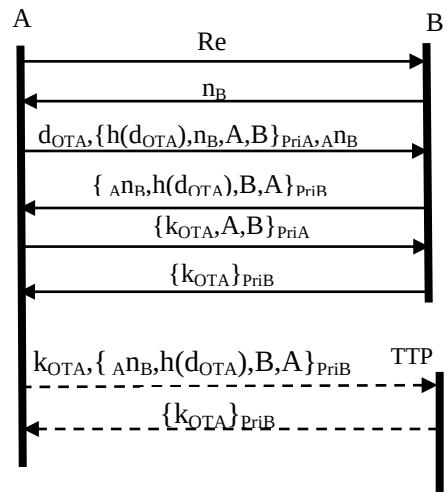
Proof basis	Composed Scheme	Mechanism
A1, A2,A3 Inv1, Inv2, Inv3, Inv4, HEA		*B* can prove integrity, authenticity and recency of the data received as it is accompanied by a signed digest containing the hash of the data together with *B*'s nonce (sent earlier) and labels for the message originator *A* and the intended recipient *B*. *B* cannot repudiate receiving the data as *A* receives from *B* two signed messages: first made up of the hash of the encrypted data together with nonces ensuring unique run and recency, and the second with the key needed for decryption. Use of explicit labels in step 3 prevents MITM attacks (A3).

Table 3.25 Composed Scheme for Authentication, Receiver Non-Repudiation and Recency

3.6.5 Schemes for Multiple Recipients and Their Proofs

PGPS schemes for multiple recipients are generated automatically by interleaving the single recipient schemes generated in Section 3.6.4. The underlying generator algorithm *MProt* presented in 3.6.5.1 takes as input the sender, the sequence of recipients and the set of security assurances required. *MProt* also removes redundant elements and combines specific cryptographic elements such as nonces. Schemes for two recipients providing all combinations of security properties (except entity authentication) are listed in Table 3.26. The scheme for entity authentication property is not combined with other properties, as each entity must be authenticated before any data despatch (precondition). Section 3.6.5.2 presents the proofs for the validity of this approach.

3.6.5.1 Multiple Recipients Security Scheme Generation Algorithm

MProt(sps, s, rs) takes as argument source *s*, set of security properties *sps* and a sequence of recipients *rs* and generates a multiple recipients security scheme. It does this by interleaving the single recipient scheme for the same properties *nr* times where *nr* is the number of recipients. It is assumed that the routine *SProt(n)* returns the single recipient scheme steps (for the same properties) from source *s* to recipient *rs[n]*. The number of cycles (Terminology 3.6) in the generated *MProt* scheme is the same as the number of cycles in the *SProt* scheme. Each complete cycle in *MProt* will have *nr+1* steps formed interleaving the messages sent to the recipients and the messages received by the source. If the *SProt* for given security properties ends with an incomplete cycle the *MProt* scheme too will end with an incomplete cycle with *nr* steps, formed by interleaving all the messages sent to the recipients. The main step of *MProt* is outline first, followed by the algorithm details and a running example.

The main steps of *MProt* are as follows:

1. create single recipient schemes from source to each recipient
2 form the cycles and steps of *MProt* by interleaving the *Sprot(i)* as *i* varies from *1* to *nr*
3 remove the repeated elements and combine other elements which can be safely composed.

Algorithm Details:

1. Create the single recipient schemes *SProt(n)* for given properties *sps*, source *s* and *rs[n]* where *n* is the index into the sequence of recipients *rs*

2. The message contents in *MProt* is formed by interleaving corresponding cycles of *SProt* as follows:

   For each complete cycle j in *MProt*: the i^{th} step is computed as follows:

   $$\sum_{k=i}^{nr} \text{message sent from source in } j\text{th cycle of } SProt(k)$$

   $$+ \sum_{k=1}^{i-1} \text{message responses to source from } j\text{th cycle of } SProt(k)$$

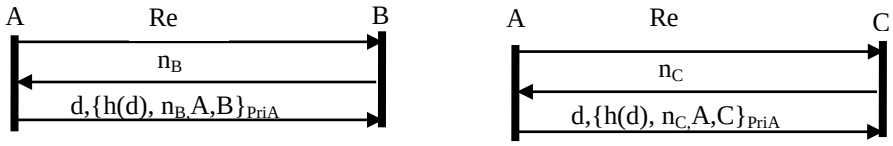
For an incomplete cycle *j* in *MProt*: the i^{th} step is computed as follows:

$$\sum_{k=i}^{nr} \text{message sent from source in } j\text{th cycle of } SProt(k)$$

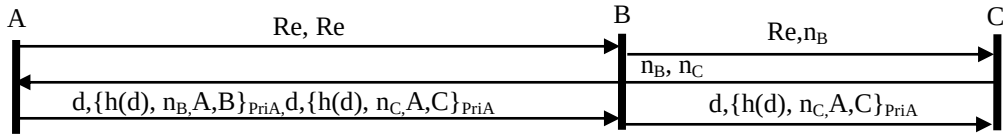
3. Repeated elements are removed, and specific elements are combined to reduce the number of terms. Each recipient combines its own nonce with the earlier one to form a composite nonce as in $n_B + n_C = n_{BC}$.

Running Example using *MProt({A,R}, A, [B,C])*

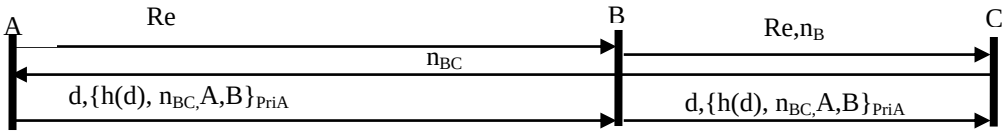
1. For the running example: *s* = *A*, rs = [*B,C*] and *sps = {A,R}* and the security schemes for *A* to *B* and *A* to *C* are as shown below:



2. For the running example, with one complete and one incomplete cycles, the message contents would be as shown in the scheme below. Note some of the terms (*Re, d*) are repeated.



3. For the running example, repeated terms are (*Re, d*) are removed and the nonces n_B, n_C are combined to form n_{BC}, which help reduce the number of distinct encryptions required. The resulting simplified scheme is shown below.



Schemes for All Basic/Composite Security Properties Generated for Two Recipients

Table 3.26 lists the schemes generated automatically for data sent from A to recipients B and C for all combinations of security properties (entity authentication is not combined with others as it is used on its own). The table uses the following symbols/functions.

d ≡ data	*h(d)* ≡ hash of *d* (data)
kOTA ≡ one-time key used by *A*	*dOTA* ≡ d encrypted *A*'s one-time key kOTA
${}_A n_{BC}$ ≡ *A*'s nonce to *B* and *C*	n_{BC} ≡ The nonce formed combining n_B with n_C
Re ≡ Symbol for recency	*EA* ≡ Symbol for entity authentication

EA	subS(A,{B,C}, {EA}, null) = A[EA, nA]B • B[EA,nA,{nA} PriB]C • C[{nA} PriB,{nA} PriC]A
A	subS(A,{B,C}, {A},d) = A[d, {h(d),A,(B,C)}PriA]B • B[d,{h(d),A,(B,C)}PriA]C
DI	subS(A,{B,C}, {DI},d) = A[d,{h(d),A}PriA]B • B[d,{h(d),A}PriA]C
RNR	subS(A,{B,C},{RNR}, d) = A[dOTA,AnBC,]B • B[dOTA, AnBC,{AnBC,h(dOTA),B,A}PriB]C • • C[{AnBC,h(dOTA),B,A}PriB, {AnBC,h(dOTA),C,A}PriC]A •A[{kOTA,A,(B,C)}PriA] B • B[{kOTA,A,(B,C)}PriA, {kOTA}PriB] C• C[{kOTA}PriB, {kOTA}PriC]A
R	Subs(A,{B,C}, {R}, d) = A[Re]B • B[Re,nB]C • C[nBC]A •.A[d,nBC]B • B[d,nBC]C
A,DI	subS(A,{B,C}, {A,DI},d) = A[d, {h(d),A,(B,C)}PriA]B • B[d,{h(d),A,(B,C)}PriA]C
A,RNR	subS(A,{B,C}, {A,RNR},d) = A[dOTA,{h(dOTA),A,(B,C) }PriA,AnBC] B • B[dOTA,{h(dOTA),A,(B,C) }PriA, AnBC,{ AnBC,h(dOTA),B,A}PriB]C • C[{ AnBC,h(dOTA),B,A}PriB,{ AnBC,h(dOTA),C,A}PriC] A • A[{kOTA,A,(B,C)}PriA] B • B[{kOTA,A,(B,C)}PriA, {kOTA}PriB] C• C[{kOTA}PriB, {kOTA}PriC]A
A,R	subS(A,{B,C}, {A,R},d) = A[Re]B • B[Re,nB]C • C[nBC]A•A[d,{h(d),nBC,A,(B,C)}PriA]B • B[d,{h(d),nBC,A,(B,C)}PriA]C
DI,RNR	subS(A,{B,C}, {DI,RNR}, d) = A[dOTA,{h(dOTA),A }PriA, AnBC] B • B[dOTA,{h(dOTA),A}PriA, AnBC,{ AnBC,h(dOTA),B,A}PriB]C • C[{ AnBC,h(dOTA),B,A}PriB,{ AnBC,h(dOTA),C,A}PriC] A •A[{kOTA,A,(B,C)}PriA] B • B[{kOTA,A,(B,C)}PriA, {kOTA}PriB] C• C[{kOTA}PriB, {kOTA}PriC]A
DI,R	subS(A,{B,C}, {DI,R},d) = A[Re]B • B[Re,nB]C • C[nBC]A•A[d,{h(d),nBC,A}PriA]B • B[d,{h(d),nBC,A}PriA]C
RNR,R	subS(A,{B,C}, {R,RNR}, m) = A[Re]B • B[Re,nB]C • C[nBC]A . A[dOTA,AnBC, {nBC,h(dOTA)}PriA]B • B[dOTA, AnBC, {nBC,h(dOTA)}PriA,{AnBC, nBC,h(dOTA),B,A}PriB]C • C[{AnBC, nBC,h(dOTA),B,A}PriB, {AnBC,h(dOTA),C,A}PriC]A •A[{kOTA,A,(B,C)}PriA] B • B[{kOTA,A,(B,C)}PriA, {kOTA}PriB] C• C[{kOTA}PriB, {kOTA}PriC]A
A,DI,RNR	subS(A,{B,C}, {A,RNR},d) = A[dOTA,{h(dOTA),A,(B,C) }PriA,AnBC] B • B[dOTA,{h(dOTA),A,(B,C) }PriA, AnBC,{ AnBC,h(dOTA),B,A}PriB]C • C[{ AnBC,h(dOTA),B,A}PriB,{ AnBC,h(dOTA),C,A}PriC] A • A[{kOTA,A,(B,C)}PriA] B • B[{kOTA,A,(B,C)}PriA, {kOTA}PriB] C• C[{kOTA}PriB, {kOTA}PriC]A
A,DI,R	subS(A,{B,C}, {A,DI,R},d) = A[Re]B • B[Re,nB]C • C[nBC]A•A[d,{h(d),nBC,A,(B,C)}PriA]B • B[d,{h(d),nBC,A,(B,C)}PriA]C
A,RNR,R	subS(A,{B,C}, {A,RNR,R},m) = A[Re]B • B[Re,nB]C • C[nBC]A •A[dOTA,AnBC,{h(dOTA),nBC,A,(B,C)}PriA]B • B[dOTA,AnBC,{h(dOTA),nBC,A,(B,C)}PriA, {AnBC,h(dOTA),B,A}PriB]C • C[{AnBC,h(dOTA),B,A}PriB, {AnBC,h(dOTA),C,A}PriC]A • A[{kOTA,A,(B,C)}PriA] B • B[{kOTA,A,(B,C)}PriA, {kOTA}PriB] C• C[{kOTA}PriB, {kOTA}PriC]A
DI,RNR,R	subS(A,{B,C}, {DI,RNR,R},m) = A[Re]B • B[Re,nB]C • C[nBC]A •A[dOTA,AnBC,{h(dOTA),nBC,A}PriA]B • B[dOTA,AnBC,{h(dOTA),nBC,A}PriA, {AnBC,h(dOTA),B,A}PriB]C •C[{AnBC,h(dOTA),B,A}PriB, {AnBC,h(dOTA),C,A}PriC]A •A[{kOTA,A,(B,C)}PriA] B • B[{kOTA,A,(B,C)}PriA, {kOTA}PriB] C• C[{kOTA}PriB, {kOTA}PriC]A
A,DI,RNR,R	subS(A,{B,C}, {A,DI,RNR,R},m) = A[Re]B • B[Re,nB]C • C[nBC]A •A[dOTA,AnBC,{h(dOTA),nBC,A,(B,C)}PriA]B • B[dOTA,AnBC,{h(dOTA),nBC,A,(B,C)}PriA, {AnBC,h(dOTA),B,A}PriB]C •C[{AnBC,h(dOTA),B,A}PriB, {AnBC,h(dOTA),C,A}PriC]A •A[{kOTA,A,(B,C)}PriA] B • B[{kOTA,A,(B,C)}PriA, {kOTA}PriB] C• C[{kOTA}PriB, {kOTA}PriC]A

Table 3.26 Schemes Generated from A to [B,C] using Different Combination of Properties

3.6.5.2 Multiple Recipients Security Schemes: Proofs

Strand Spaces [106] are used in this section to prove that multiple recipient schemes (protocols) created by interleaving single recipient schemes are secure. The Dolev-Yao threat model [102] used assumes that the communication medium is under the control of the adversary who is free to block, readdress, fake or duplicate messages. The adversary starts with a set of initial keys and acquires new knowledge by intercepting the messages exchanged. The adversary can send fake messages of its own using keys in its possession. Intruder capabilities presented in Table 3.27 models the power of the adversary explicitly, which are proved to be inadequate to subvert the protocol goals for various properties. All the proofs in this section make the **perfect encryption assumption** defined in Section 2.1.2. The proofs in this section follow the Strand Spaces methodology presented in Section 2.2.1.2. In addition, this section uses the honest entity assumption which states honest entities do not disclose private keys and the assumption that all nonces (random numbers) are freshly generated (Inv1 and Inv2 in Table 3.12).

Assumed Intruder Capabilities Modelled as Adversary Strand

The set of initial knowledge of the adversary strand is represented by K_A. This contains the public keys of all parties and the symmetric keys the adversary shares with other parties. The capabilities of the adversary strand to deduct new facts are assumed to be those listed in Table 3.27.

Strand Capability	Operation	Description
M[t]	$<+t>$	Inserts *t* where $t \in K_A$
G[g]	$<-g>$	Flushing (blocking)
T[g]	$<-g, +g, +g>$	Tee (forwarding term along two other strands)
C[g,h]	$<-g, -h, +g.h>$	Concatenation of two terms
R[g,h]	$<-g.h, +g, +h>$	Separation of term
K[k]	$<+k>$ *where* $k \in K_A$	Sending a key in possession
E[k,h]	$<-k,-h, +\{h\}_K>$	Sending an Encrypted term
D[k,h]	$<-k^{-1}, -\{h\}_K, +h>$	Sending a term after decrypting it

Table 3.27 Capabilities of Adversary

Authentication Property

The authentication property in PGPS is based on the aliveness property, which requires proving that "*whenever data is received by entity B from a trusted entity A, that data must have been sent by A to B in an earlier event*". Stronger assurances amounting to "*injective agreement*" requires combining the authentication property with recency [33].

Strand Space for Multiple Recipient Authentication Scheme

In the multiple recipient scheme all authenticated data are signed, i.e. accompanied by a digest encrypted with the private key of the sender, which contains a hash of the data, data origin and all data

destinations. This scheme therefore requires proving that data signed by a valid sender can only originate in one specific entity, the entity of data origin. The diagram below shows a Strand Space bundle with a number of intermediaries preceding *Rn*, which can be either adversaries or other protocol recipients. A Strand Space bundle for this scheme is shown in Figure 3.12.

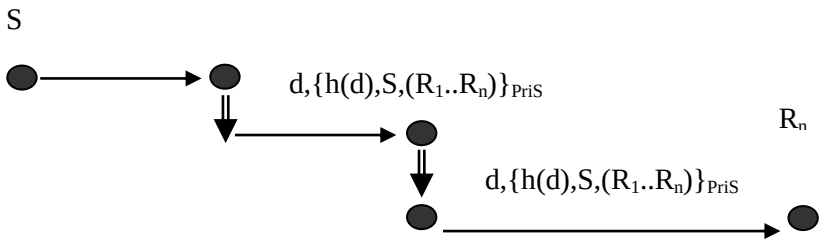


Figure 3.12 Strand Spaces Bundle for Authentication Scheme

Assumption: Dolev-Yao Threat Model and Intruder Capabilities as shown in Table 3.27.

Proof of Authentication Scheme for Multiple Recipients: Requires proving each data recipient R_i .. R_n can be assured that data originated in source *S* and was directed to it**.**

The data *d* received together with the digest containing source and recipients above must have originated in *S*, as tampered data will not produce the same hashed value as that received in the term signed by originator *S*. The presence of recipient label R_1 ..R_n (the intended destinations) within the term signed by *S* can be used to prove the data was directed to the recipients R_1 ..R_n. In order for an adversary to introduce the encrypted term $\{h(d),S,(R_1..R_n)\}_{PriS}$ into the strand using its capability *E[k,h]* in Table 3.27, the strand must receive the key pri_S in the open. However, the invariant **Inv1** (Table 3.12) precludes honest entity *S* from sending its private key in the open**.** An adversary can replay such a message at a later time using its capability *T[g]* or from known terms using the capability *M[t]*.However, authentication property is not required to prevent replay attacks; authentication must be combined with the recency in PGPS to prevent such attacks.

Data Integrity Property

Strand Space for Multiple Recipient Data Integrity Scheme

The scheme used is identical to authentication shown in Figure 3.12 except that it has no label for the recipient (refer to Section 3.6.3).

Assumption: Dolev-Yao Threat Model and Intruder Capabilities as shown in Table 3.27.

Proof of Data Integrity Scheme for Multiple Recipients: Requires proving data received by each recipient R_i .. R_n can be assured that data from source S has not been tampered**.**

The data *d* received together with the digest must have originated in *S*, as private keys are never divulged by honest parties (**Inv1**) (table 3.12) and different data will not produce the same hashed value.

Receiver Non-Repudiation Property

Receiver non-repudiation schemes work by including additional evidence in an acknowledgement sent, allowing a third party to verify data receipt. This assurance incorporates an "*injective agreement*", which requires a one-to-one correspondence between recipient and sender. The PGPS non-repudiation scheme for multiple recipients adheres to this definition by getting every recipient to submit evidence of data receipt signed by its own private key. This signed evidence incorporates a nonce from the sender (that can be generated by a third party if necessary) to prove a unique run.

Strand Space for Multiple Recipient Receiver Non-Repudiation Scheme

The scheme used by this property facilitates fair exchange by providing access to data in its final form only after receiving acknowledgement for a partial release. In the first pass, the encrypted data d_{OTS} is sent together with a nonce ${}_{S}n_{Rs}$. The nonce sent from *S* to all the recipients in the group R_s can be generated by the trusted third party if necessary (before sending data that requires non-repudiable evidence of message receipt) to provide stronger evidence of a unique run. The one-time key needed for decryption is released only after receiving a signed digest containing a hash of the previously sent partial data, nonce and labels of the recipient and sender. The scheme ensures that the one- time key is released only after receiving all the acknowledgements. This scheme, therefore, requires proving that the acknowledgement came from a specific recipient and is associated with a unique send event identified by the nonce. Figure 3.13 shows a Strand Space bundle with a number of intermediaries between sender *S* and a particular recipient *Rn*, which can include both adversaries and other valid protocol recipients.

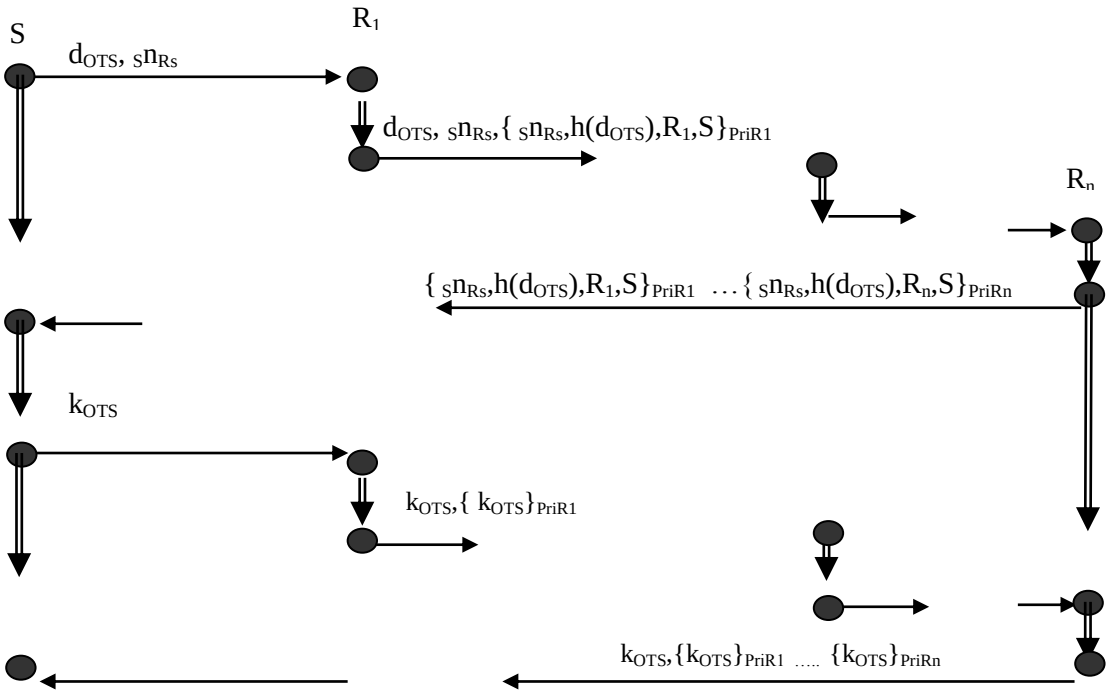


Figure 3.13 Strand Spaces Bundle for the Non-Repudiation Scheme

Assumption: Dolev-Yao Threat Model and Intruder Capabilities as shown in Table 3.27.

Proof of Non-Repudiation Scheme for Multiple Recipients: Requires proving none of the recipients R_1 .. R_n can deny receiving the data from source *S*.

The digest received by *S* containing the nonce, recipient label and the hash of encrypted data from each of the recipients must be the acknowledgement from R_1 .. R_n, as private keys are never divulged by trusted intermediaries (**Inv1** in Table 3.12). Also, R_1 .. R_n must be in possession of *d* as only identical data can produce an identical hash value. The presence of recipient label (one of R_1 .. R_n and *S*) prevents type-flaw attacks; no adversary strand can create such a digest using it capability *E[k,h]* (in Table 3.27) without the private keys of R_1 .. R_n. The signed acknowledgement for K_{OTS} by R_1 .. R_n can be used to prove it had received the one-time key necessary to retrieve the data. The scheme is fair, as the one-time key for decrypting the data will be released only after verifying the acknowledgements for all recipients R_1,…R_n.

Recency Property

The recency property allows data recipients to place an implicit lower bound on the time of data despatch by comparing it to an earlier event. It cannot, however, guarantee data origin as it cannot prevent attacks by adversaries. Recency property must be combined with authentication property to prove data origin. Recency property must be combined with non-repudiation property if guaranteed timely data delivery is required.

Strand Space for Multiple Recipient Recency Scheme

The scheme requires the concatenation of nonces generated by recipients allowing the last valid recipient to send the composite nonce to the sender. The sender attaches the data to the composite nonce. The recipients can extract their own nonce to verify that the data received has been recently generated by the sender. This scheme requires each recipient to show that the data was sent after an earlier event. This property however, does not assure that the data was generated by a specific source entity. If such an assurance is required recency must be combined with authentication. Figure 3.14 shows a Strand Space bundle with a number of intermediaries between sender *S* and a particular recipient *Rn*, which can include both adversaries and valid protocol recipients.

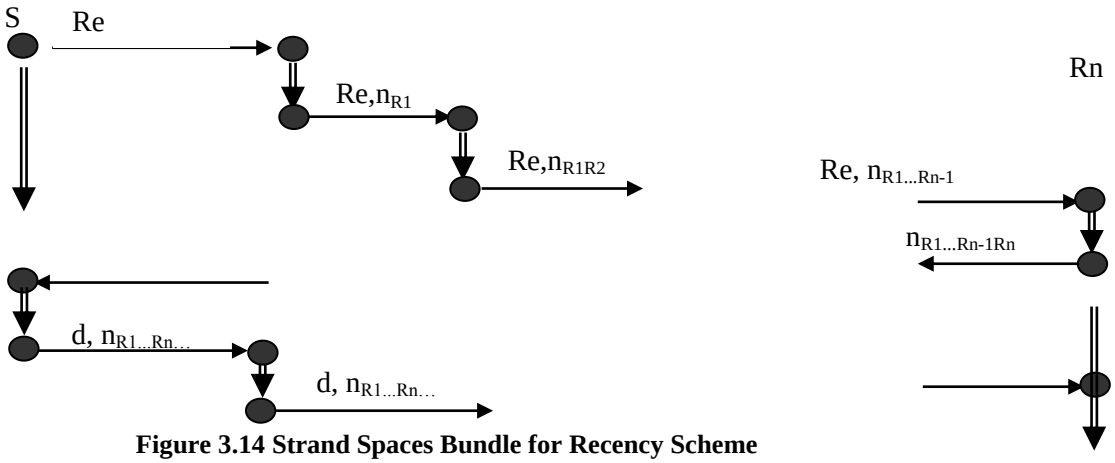


Figure 3.14 Strand Spaces Bundle for Recency Scheme

Assumption: Dolev-Yao Threat Model and Intruder Capabilities as shown in Table 3.27.

Proof of Recency Scheme for Multiple Recipients: Requires proving all of the recipients can be assured that the data received is recent.

The message received by Rn containing its own recent nonce with data must have been sent recently as nonces are assumed to be unique and freshly generated (Inv2).

3.6.6 PGPS Performance

E-commerce protocols cannot be selected on the basis of security alone, because high security-strength components can impact on the performance, including response time, line costs and throughput [45]. For example, maximum throughput of SSL can vary by a factor of five, depending on the cryptographic algorithms and the key lengths used [49]. The performance impact is even greater for resource limited devices [52]. Finding the right trade-offs between security and performance requires modelling the cost in terms of underlying cryptographic elements [45, 107, 108]. Performance measures including response-time, throughput and communication costs are modelled indirectly in PGPS based on underlying factors, protocol-bandwidth and computational cost. The computational-cost is expressed in terms of the number of CPU cycles required to perform the cryptographic operations. It is based on the amount of cryptographic operations carried out at the data originator and recipients. The PGPS cost model assumes a flexible architecture where security requirements are set using fine-grained security levels with varying security strengths.

3.6.6.1 Analysis of PGPS Protocol Generation Algorithm

If protocols (schemes) for multiple recipients are to be generated dynamically the underlying algorithm should scale well. Hence, this section computes how the total-number-of-terms vary with increasing number of recipients. The total number of terms including piggybacked elements is used because the underlying algorithm needs to process those terms (removing redundancies, merging terms). The total number of terms in turn reflects the number of new terms added for each additional recipient, which varies based on the schemes used by different properties. Hence, the number of steps and size of data are modelled as a function of the number of recipients (*nR*) and the underlying schemes used.

Total Number of Steps

The number of cycles (*nC*) in *MProt* is determined by the number of cycles in the *SProt*. An upper bound on the number of cycles in *SProt*: *nC = nS mod 2 + 1* where *nS* is the number of send and receive events as two such events form a complete cycle (Terminology 3.6). As outlined in the algorithm, when *nR* is the number of recipients there can be at most *nR+1* steps. Therefore, an upper bound on the total number of steps is: *totSteps = (nS mod 2 + 1) *(nR + 1)*. Thus, for any scheme with *nS* send-and-receive events (fixed), the total number of steps (*totSteps*) grows linearly with *nR*.

Total Number of Terms

The total number of terms is estimated based on the number of new terms introduced by each additional recipient. In Table 3.28 the first row shows the number of new terms introduced when $(n+1)^{th}$ recipient is added, while the second row shows the total terms after n recipients. For example, the data integrity scheme for an additional recipient introduces only one additional step with two terms (the data and the signed digest) and requires no changes to any of the other steps (the previous recipient simply forwards the data and the digest from the source entity). Schemes for all other properties require some changes to either steps used in forwarding the message from the source, or the steps used in carrying the responses back to the source, or both.

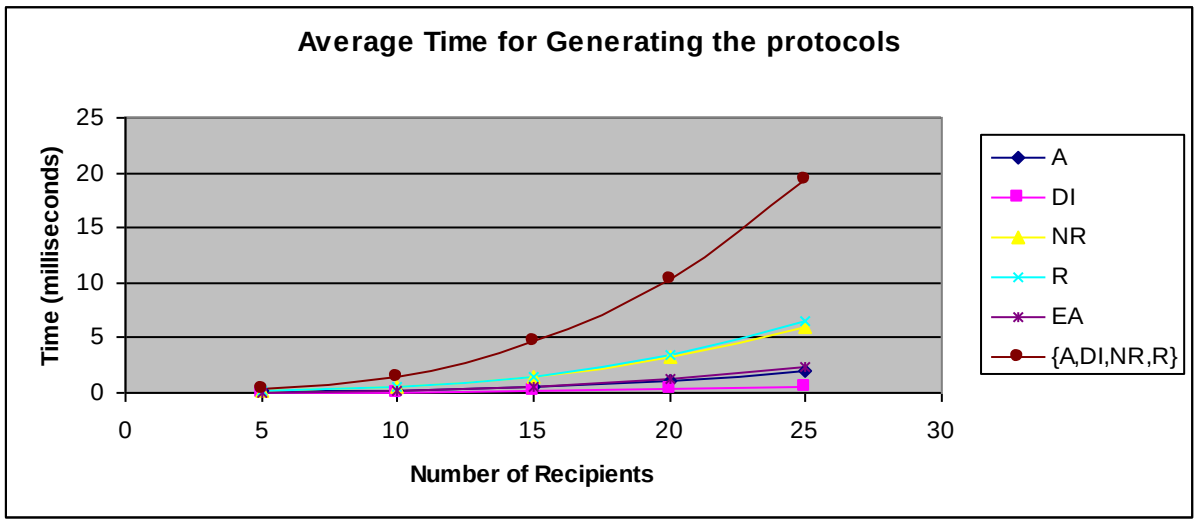
	DI	**A**	**EA**	**R**	**RNR**	**{A,DI,R,RNR}**
New terms for $(n+1)^{th}$ recipient.	2	$n+1$	$n+2$	$2n+2$	$2n+3$	$5n+6$
Total number of terms after n recipients	$2n$	$\frac{n(n+3)}{2}$	$\frac{n(n+5)}{2}$	$n(n+3)$	$n(n+4)$	$\frac{n(5n+7)}{2}$

Table 3.28 The Number of New Terms introduced for N^{th} Recipient and the Total Number of Terms

This analysis shows that the performance for *DI* property in linear, as the number of terms for *n* recipients is only *2n*. The schemes for all other properties is order $O(n^2)$. In the next section, the result of this algorithm analysis is corroborated by the measured performance.

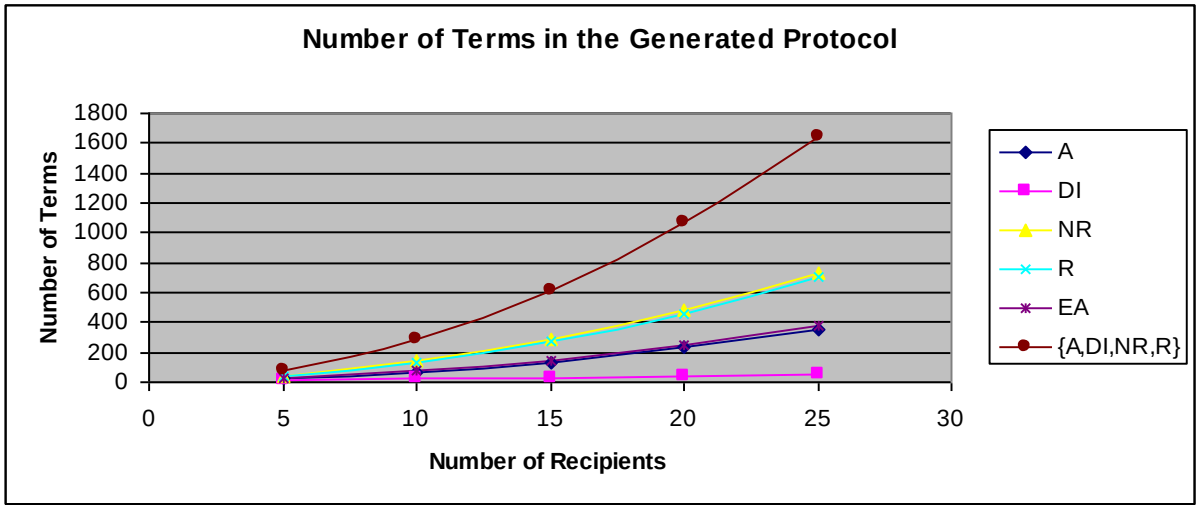
3.6.6.2 Performance of the Protocol Generation Algorithm

This section presents the results of running the protocol generation algorithm varying the number of destinations and the security properties. The response time was measured running the algorithm with a 2.5 GHz laptop with 4 GB RAM and Windows XP operating systems. The algorithm was implemented as a Java program with 750 lines of code. The elapsed time was obtained based on the average of 10,000 runs (to reduce the impact of background processes running). Figure 3.15 shows how the elapsed time varies with the number of recipients while Figure 3.16 shows how the number of protocol terms increases with the number of recipients.



A – Authentication R – Receny EA – Entity Authentication DI – Data Integrity NR–Receiver Non-Repudiation

Figure 3.15 Protocol Generation Time for Varying Number of Recipients



A – Authentication R – Receny EA – Entity Authentication DI – Data Integrity NR–Receiver Non-Repudiation

Figure 3.16 Number of Protocol Terms for Varying Number of Recipients

Analysis

The analysis attempts to evaluate algorithm complexity in terms of processing time. Processing time is modelled using total number of terms, as each additional term involves processing. Total number of terms for *n* recipients is derived by summing the new terms introduced by each additional recipient (refer to Table 3.28). Scheme for *{DI}* involves *2n* terms, thus resulting in linear complexity. The scheme for {A,DI,RNR,R} involves $\frac{n(5n+7)}{2}$ terms, thus resulting in quadratic complexity. Schemes for {RNR}, { A } and { R } also shows quadratic complexity as the numbers of terms involved are $n(n+4)$, $\frac{n(n+3)}{2}$ and $n(n+3)$ respectively. It is evident from Figure 3.15 and 3.16 the analytical results corroborate well with experimental results, with the graphs for all the properties showing very similar rates of increase.

The experimental results show that elapsed time is less than 20 ms even for the scheme generated for up to 25 recipients with the combination of all security properties {A,DI,RNR,R}. This shows the viability of the method presented as the number of recipients is unlikely to exceed 25, in most practical applications.

3.6.6.3 Security Overheads of PGPS Generated Schemes

One past approach uses the state of commitment resulting from message exchanges as the basis for protocol synthesis [94]. Such commitments can be made enforceable if such messages are exchanged with the right PGPS security levels. Different combinations of messages may be used to arrive at the desired state of commitment; however each message may require a different security level that reflects the underlying semantics. For example, in Figure 3.17-A customer *C* sends a quote request message (*QReq*) to merchant *M* which responds with a quote response message (*QRes*) with security property {*A*} which includes the product, price, quantity and timeframe. Finally the customer *C* sends a purchase order (*PO*) which includes the product, price, quantity and timeframe with security properties {*A,RNR*}. At the end of these message exchanges the customer is committed to pay when goods are delivered and the merchant is committed to supply at the stated price and timeframe. If however, customer *C* has an established relationship with *M* which requires it to supply goods at pre-set price and timeframe, both *B* and *C* may reach the same level of commitment by sending a different purchase order (*PrePO*) message which includes only the product and quantity as shown in Figure 3.17-B. In such settings the candidate protocol with the least overall protocol cost can be the basis for selection. The next section expresses the overall protocol bandwidth and computational cost as a function of underlying security strength.

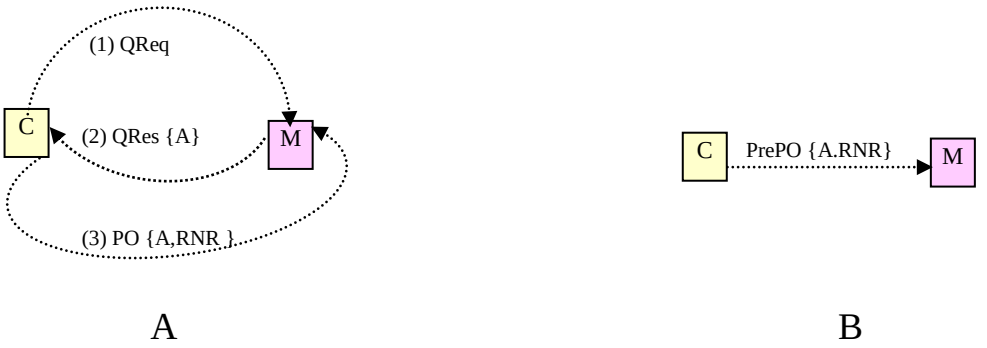


Figure 3.17 Two Different Message Exchanges Producing the Same Outcomes

3.6.6.4 Computational Cost

The computational cost incurred by protocols can be used as a means to select the optimal protocol when multiple protocols meet the security requirements. The overall protocol computational cost (*PC*) in PGPS can be expressed as the sum of sub-protocol costs (SPC_i), based on the schemes devised in 3.6.5 for sending an individual message for multiple recipients as in

$$PC = \sum_{i=1}^{l} SPC_i$$

where $SPC(M_i)$ is the sub-protocol cost for the i^{th} message, and l is the number of messages. The cost of each sub-protocol scheme is expressed as a function of protocol security strength *ss* (refer to Section 2.6) and message security-level *sl* (Terminology 3.3). Sub-protocol cost is based on the sum of costs for non-correspondence (*NCP*) properties (such as secrecy) enforced at the data element level and costs for correspondence (*CPC*) properties (such as authentication, integrity) enforced at the data level.

$$SPC_i = SPC\_NCP_i(ss) + SPC\_CP_i(ss,sl)$$

The cost of enforcing secrecy for each data element is based on the cost of encryption at the source before other sub-protocol operations, and the cost of decryptions at destination entities afterwards. In PGPS secrecy cost is expressed in terms of the number of secret data elements in i^{th} sub-protocol (NSE_i), the average size of secrecy group in i^{th} sub-protocol ($SecretGrpSize_i$) and security strength (*ss*).

$$SPC\_NCP(M_i, ss) = CostSK(NSE_i, SecretGrpSize_i, ss)$$

The cost of enforcing a security level *sl* using security strength *ss* is the sum of costs for enforcing the individual properties *sp* constituting *sl* as shown below.

$$SPC\_CP_i(ss, sl) = \sum_{j=1}^{n} SPC\_CP_i(ss, sp_j) \text{ where } sl = \bigcup_{j=1}^{num} sp_j$$ and *num* is the number of basic properties in *sl*. The cost of enforcing individual correspondence properties is based on the cost incurred at each step of the protocol, which is security level and security strength dependent. These costs are stated using the functions described in the Table 3.29.

Function and Variables	Meaning	Arguments to functions
SN(ss)	Size of nonce	security strength
CN(ss)	Cost of nonce	security strength
CPrE(ss,size)	Cost of private key encryption	security strength, size of data
SPrE(ss,size	Size of private key encryption	security strength, size of data
CPuD(ss,size)	Cost of public key decryption	security strength,size of data
SPuD(ss,size)	Size of public key decryption	security strength,size of data
COTKE(ss,size)	Cost of one-time-key encryption	security strength,size of data
COTKD(ss,size)	Cost of one-time-key decryption	security strength,size of data
SOTKE(ss,size)	Size of one-time key encryption	security strength,size of data
SOTK	Size of one-time key	
SH(ss,size)	Size of hash	security strength, size of data
CH(ss,size)	Cost of hash	security strength, size of data
$n = NR_i$	Number of recipients in i^{th} sub-protocol	
NSE_i	Number of secret elements in i^{th} sub-protocol	
SD_i	Size of data in i^{th} sub-protocol	
NMH	Number of message headers	
HS	Header size	

Table 3.29 Functions used for Describing Protocol Costs

Table 3.30 shows estimated computational costs for i^{th} sub-protocol sent to *n* recipients with elementary properties entity authentication (EA), data integrity (DI), authentication (A), recency (R) and receiver non-repudiation (RNR).

$$SPC\_CP_i(ss, EA) = n.(C\Pr E(ss, SN(ss)) + CN(ss) + n.(CPuD(ss, SN(ss))$$

$$SPC\_CP_i(ss, R) = n.CN(ss)$$

$$SPC\_CP_i(ss, RNR) = COTKE(ss, SD_i) + n.COTKD(ss, SOTKE(SD_i)) + n.(C\Pr E(SOTKE(ss, SD_i)))$$
$$+ CN(ss) + n.(CPuD(ss, SOTKE(ss, SD_i))) + n.(C\Pr E(ss, SOTK(ss)) + n.(CPuE(ss, SOTK(ss)))$$

$$SPC\_CP_i(ss, DI) = n.(C\Pr E(ss, SH(ss, SD_i) + 1)) + n(CPuD(ss, SH(ss, SD_i) + 1)$$
$$+ (n+1).CH(ss, SD_i)$$

$$SPC\_CP_i(ss, A) = n.(C\Pr E(ss, SH(ss, SD_i + n/2)) + n(CPuD(ss, SH(ss, SD_i + n/2))$$
$$+ (n+1).CH(ss, SD_i)$$

Table 3.30 Computational Cost for Multiple Recipient Protocols

3.6.6.5 Protocol Bandwidth

Protocol bandwidth in PGPS measures the amount of data carried from one point to another and is expressed as a function of message sizes, the number of recipients, the protocol security strength, the size of message headers and the number of hops. The number of message headers (*NMH*) for a given security level can be derived directly from the cryptographic schemes associated with that security level. The Table 3.31 shows the *NMH* values for n recipients in PGPS. Note the number of hops required for the security levels *{A}* and *{DI}* is the same as that required for *{A,DI}*.

A	DI	RNR	R	EA	A,DI	A,RNR	A,R	A,R,DI	A,R,RNR	A,DI,RNR	R,DI,RNR	A,R, RNR,DI
n	n	2n+1	2n+1	n+1	n	2n+1	2n+1	2n+1	3n+2	2n+1	3n+2	3n+2

Table 3.31 Number of Headers Used for Security Levels

Protocol bandwidth in PGPS is computed by summing up the bandwidths of all sub-protocols. The number of hops used for security levels with multiple properties is usually less than the sum for individual properties, as some hops are combined. The bandwidth for headers is therefore expressed as a function of the sub-protocol security level while the bandwidth for cryptographic components is expressed as the sum of the parts of individual properties.

$$PB(M) = HS.\sum_{i=1}^{l} NMH(n, sl_i) + \sum_{i=1}^{l} SD_i.n_i + \sum_{i=1}^{l} SPB_i$$ where l is the number of sub-protocols.

Sub-protocol bandwidth (*SPB*) is expressed as the sum of bandwidth resulting from non-correspondence (*SPB_NCP*) properties (such as secrecy) and correspondence properties (*SPB_PC*) at message level.

$$SPB_i = SPB\_NCP_i + SPB\_CP_i(ss,sl)$$

The additional bandwidth for enforcing secrecy can be expressed as a function of tag size (*TAGSIZE*), number of secret elements (*NSE*) and number of destinations (*ND).*

$$SPB\_NCP_i = TAGSIZE\ .\ n\ .\ NSE_i$$

Each correspondence property making up the security level causes an increase in bandwidth through additional cryptographic elements as shown below.

$$SPB\_CP_i(ss, sl) = \sum_{j=1}^{num} SPB\_CP_i(ss, sp_j) \qquad \text{where } sl = \bigcup_{j=1}^{num} sp_j$$

Table 3.32 shows protocol bandwidth for i^{th} sub-protocol sent to n recipients, for each of the correspondence properties.

$$SPB\_CP_i(ss, EA) = n.SD_i + 2.n.SL(ss) + n.(n+1).SPuK(ss, SN(ss))/2$$

$$SPB\_CP_i(ss, A) = n.SD_i + n.SPuK(n.SL(ss) + SH(ss, SD_i))$$

$$SPB\_CP_i(ss, DI) = n.SD_i + n.(SL(ss) + SH(ss, SD_i))$$

$$SPB\_CP_i(ss, RNR) = n.(SOTKE(ss, SD_i) + SN(ss)) + n.(n+1).(SPuK(ss, 2.SL(ss) + SN(ss) + SH(ss, SD_i)) + n.(SPuK(n.SL(ss) + SOTK(ss) + n.(n+1).S\,\mathrm{Pr}\,K(ss, SN(ss) + 2.SL(ss))$$

$$SPB\_CP_i(ss, R) = n.SD_i + n.(n+1).SN(ss)/2 + n.SL(ss))$$

Table 3.32 Bandwidth for Multiple Recipient Protocols

Expressing computational cost and bandwidth as a function of security strength makes trade-offs between security and performance possible, by selecting the right security strength. Furthermore when two or protocols meet the requirements the optimal one can be selected on the basis of cost.

3.7 Applicability: Dynamic Generation of Security Protocols

One main benefit of the PGPS approach is the ability to generate security protocols at runtime which is vital in e-commerce and web services. However, any protocol generated must meet both functional and security requirements of all interacting entities. These requirements in turn vary with the underlying domain, the role of interacting entities and the semantics of data elements. This section presents a technique that allows such constraints to be specified in terms of data elements and underlying classes or roles with predefined meanings. Section 3.7.1 explains how a knowledge based approach can be used to automate the synthesis process. Section 3.7.2 compares the performance of PGPS with other approaches.

3.7.1 Automated Synthesis (SYN)

This section describes the technique devised to automate protocol synthesis (SYN) based on requirements stated in terms of knowledge elements in the initial and final worlds. These requirements together with knowledge dependencies and semantic constraints specified form the basis for computing the necessary message transitions with the associated security levels. The messages made up of one or more knowledge elements are formed to transform the entity knowledge level from specified initial state to the required final state without violating the security requirements. End-to-end security requirements are met based on security properties specified explicitly for knowledge elements in the initial and final worlds and implicitly for knowledge element dependencies. Together, they help meet the end-to-end security requirements, vital in e-commerce. For example, if a customer requests a

service from a broker with a time-bound property in a specific domain the broker may be required to interact with other service providers and respond using the same time-bound to meet the end-to-end security requirements. The message security levels necessary to meet such end-to-end security requirements can be enforced using the schemes presented in Section 3.6. Section 3.7.1.1 describes the model elements formally while section 3.7.1.2 describes the synthesis process using an example.

3.7.1.1 Model Elements and the Basis for Synthesis

TERMINOLOGY 3.7: MODEL ELEMENTS

- *E represents the set of all trading entities and intermediaries in the domain*
- *ER represents the set of all entity roles such as Bank, Merchant or Customer.*
- *KE represents a knowledge element such as the customer-credit-information.*
- *KL (knowledge level) for an entity represents the set of all knowledge elements currently in its possession*
- *The security requirements (SReq) for a knowledge element in initial or final state is = SC x SP*
  where SC $\subseteq$ ER be the roles given access to a specific knowledge-element and SP $\subseteq$ {A,NR,TB,DI} be the set of all correspondence security properties specified.

For example, if a knowledge-element originating in *A* can only be accessed by entities *B* and *C* using security properties authentication, *SReq* would be set to ($\{B, C\} \times \{ A\}$).

SYN models current system state assuming perfect knowledge recall, whereby all knowledge initially possessed or acquired in the past, can be recalled by entities [27]. Such a model allows a layered approach to synthesis, where knowledge of entities increase steadily until the desired goal state is reached. A layered approach allows planning graphs to be devised in a polynomial-time by limiting the number of possible transitions [46]. In this approach all intermediate system states are defined in terms of possession of knowledge elements. Entity knowledge level increases through knowledge elements received or derived. Multiple final states permit different sequences of valid actions, reflecting non-deterministic choices available to interacting entities. For example, consider a merchant entity which allows its customer entities to either rent, purchase or discontinue the trade after making initial inquiries. Possible goals can then be expressed as customer-purchasing and merchant retailing, customer-renting and merchant-hiring, or both entities suspending the trade. The entities may transition through a number of intermediate states with different knowledge levels, until one of the desired final goals is reached.

DEFINITION 3.5 KNOWLEDGE BASED SYSTEM

Let *E be the set of entities,*

S be the finite set of states,

$I \in S$ be the set of initial states,

$F \subseteq S$ be the set of final states,

SKE be the set of all knowledge-elements and

SPS the set of security properties and $T \subseteq E \times P(E) \times SKE \times SPS$ be the relation specifying the set of valid transitions.

Then the knowledge based system (KBS) incorporating security can be expressed as a tuple as in:

KBS = (E,S, I, F, T, SKE, SPS)

In a *KBS* the set of valid transitions is determined based on the current state of knowledge, dependencies and constraints specified. The current knowledge at any instance is represented by: $CK \subseteq E \times SKE$, where *E* is the set of entities and *SKE* the set of knowledge-elements.

Specifying Dependency Constraints

Dependency constraints specify new knowledge-elements that can be derived from base elements based on entity role and the underlying domain. Such a constraint may specify that a *merchant* cannot generate *supply-details* until *purchase-details*, *customer-details* and *stock-level-details* are available.

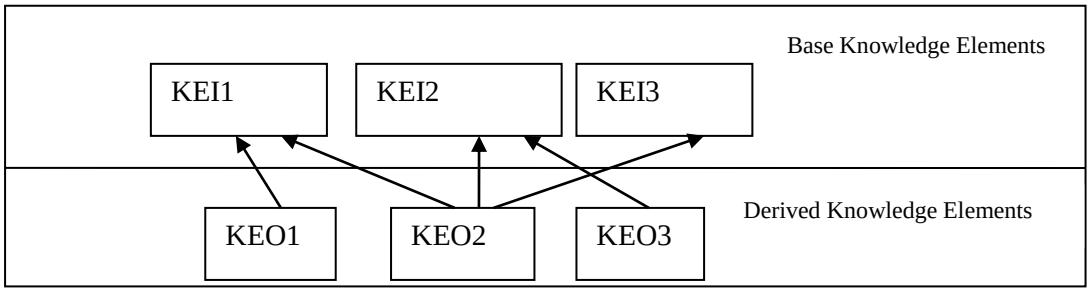


Figure 3.18 Specifying Dependenies between Knowledge-Elements

Security constraints necessary to meet end-to-end security can also be specified explicitly in terms of security properties of base and derived elements. The security properties with which a base element should be received (*RSP* constraints described in Section 6.5.3.5) can be stated in terms of security properties with which a derived element must be received. Similarly, security properties with which a derived element should be sent (*SSP* constraints also described in Section 6.5.3.5) could be made a function of security properties with which base elements were sent. Such constraints help capture the relationships that exist between the security properties and the underlying semantics in a particular domain. For example, a bank receiving an authenticated message for credit card payment from a merchant may proceed to alter the customer-balance on the premise that the merchant could only authenticate such a payment message if it had already received an authenticated message confirming goods delivery.

TERMINOLOGY 3.8: DEPENDENCY CONSTRAINTS (DC)

Let *Role be the entity role where new knowledge-elements are derived*

Base $\subseteq$ SKE be the sets of base elements

Derived $\subseteq$ SKE be the set of derived elements

Bfs be the set of functions that expresses security/trust receipt requirements of each base element as a function of receipt requirements of derived elements. These functions help propagate security/trust from message recipient to originator.

Dfs be the set of functions that expresses security/trust despatch requirements of each derived element as a function of despatch requirements of base elements. These functions help propagate security/trust from message recipient to originator.

Then the Dependency constraint DC= <Role, Base, Derived, Bfs, Dfs> specifies the dependencies for Role in terms of Base, Derived, Bfs and Dfs.

Sequencing Constraints at Entity Level

In the proposed approach a knowledge element is despatched to all other entities requiring those at the earliest time. However, the sequencing constraints (*VMD* and *VMR* below) can be used to restrict the order in which knowledge elements are despatched or received.

TERMINOLOGY 3.9:VALID MESSAGE RECEIPT (VMR)

*The predicate **VMR**(e,ke,pke,s_role) specifies e must receive all the knowledge-elements pke from an entity of with role s_role before receiving ke, where e $\in$ E, is the set of entities, ke $\in$ SKE and pke $\subseteq$ SKE where SKE is the set of all knowledge elements. The sender role or message elements can be set to null if necessary*.

For example, the predicate *VMR(Mer1, Purchase_Req, {Quote_Req}, null)* specifies merchant *Mer1* will accept the message *Purchase_Req* if it had earlier received a *Quote-Req* message from the same entity. The predicate *VMR(Mer1, Purchase_Req, {Quote_Req}, Customer)* specifies merchant *Mer1* will accept the message *Purchase_Req* from a *Customer* entity that had earlier received a *Quote-Req* message.

TERMINOLOGY 3.10: VALID MESSAGE DESPATCH (VMD)

*The predicate **VMD**(e,ke,pke,r_role) specifies all the knowledge-elements pke that an entity e must despatch to an entity of role r_role before sending the knowledge-element ke, where e $\in$ E, message element received ke $\in$ SKE , and pke $\subseteq$ SKE.*

Grouping Constraints at Entity Level

In the proposed approach all knowledge elements are despatched at the earliest time unless grouping constraints are specified reflecting the underlying semantics. Grouping knowledge elements exchanged into a single message help assure recipients that they relate to the same transaction [50]. A bank may insist on receiving a message containing encrypted information from both merchant and customer before initiating a credit-transfer between them. A merchant entity must, therefore, group

together payment and purchase details in a message before despatching it to the bank. A merchant forwarding the message elements credit-card together with purchase details cannot later refute that they were part of the same transaction. Assertions such as "elements received together are related" are made from common conventions used in standard protocol design [15]. Knowledge-elements should be grouped only when necessary as it adversely impacts performance through higher security level (set to the element with the highest security level) and delays in message despatch (until all base elements are received). SYN allow message recipients to specify grouping constraints for message receipt using VMRG. Such constraints however, can only be met if message originators permit such grouping for messages sent, using VMSG.

TERMINOLOGY 3.11: VALID MESSAGE RECEIVER GROUP (VMRG)

VMRG*(recv, megroup) is the predicate for specifying valid message receiver group where recv* $\in$ *E and megroup* $\subseteq$ *SKE.*

The predicate *VMRG(Mer1, {Purchase_Req, Customer_Info})* specifies that merchant *Mer1* will receive the message that contains both *Purchase_request* and *Customer_Info*, and either of these messages cannot be sent alone. Similarly valid message sender group *VMSG(sender, megroup)*, where *sender* $\in$ *E*, and *megroup* $\subseteq$ *SKE* specifies message elements that must be sent together.

End-to-End Security/Trust Properties Derived Through Propagation

End-to-end requirements in SYN are propagated recursively using dependency constraints expressing security requirements as a function of base and derived elements. Receiver security properties (*RSP*) are propagated form derived elements to base ones while the sender security properties (*SSP*) are propagated in opposite direction. Such an approach allows entities of data origin and destinations to influence the security level for intermediate transactions. For example in Figure 3.18, the Sender Security Property (*SSP*) for knowledge-element *KEO2* in Entity E_n is a function of the Sender Security properties (*SSP*) for its base elements as in:

$$SSP(E_n,KEO2) = fb(SSP(E_i,KEI1), SSP(E_j,KEI2), SSP(E_k,KEI3))$$

where E_i, E_j, E_k are entities from which base entities (*KEI1, KEI1, KEI1*) are despatched, and *fs* is a function over the domain of security properties (using set operators). An example of a valid function *fs* is $union(SSP(E_i,KEI1), ,SSP(E_j,KEI3),\{A\})$ which forms the resulting sender security property by combining the security properties of two of its base elements, and the security property *{A}*. Such a function (*fs*) ensures the security property is no less than *{A}* regardless of the security properties for base elements. Similarly the Receiver security property (*RSP*) for *KEI2* in E_n from which *KE02* and *KEO3* are derived acan be stated as: $RSP(E_n,KEI2) = fd(RSP(E_m,KEO2), RSP(E_n,KEO3))$.

A Message Synthesis Example Based on Knowledge Element Requirements

A simple example in this section demonstrates how constraints and requirements at the knowledge element level can be used in arriving the set messages and their security levels. Figure 3.19 shows the five knowledge elements, their dependencies, secrecy requirements and their entity of origin. For example, the knowledge element *Quote* generated based on *QReq* (quote request) and *CDet* (customer details) is kept a secret only shared by the merchant and the customer. Figure 3.20 specifies the initial and final knowledge elements for customer and merchant and their required security properties if any. For example, supply order is required in merchant with authentication property. The dependencies show the relationships between base and derived elements and the relationships between security properties. These relationships between security properties as well as the initial and final security requirements are used to arrive at the security levels for messages exchanged as shown in Figure 3.21. Note in this example all messages are despatched at the earliest possible time as no sequencing or grouping constraints are specified. Note the details of security schemes are omitted as the proven security schemes presented on Section 3.6 can be used meet the required security levels directly.

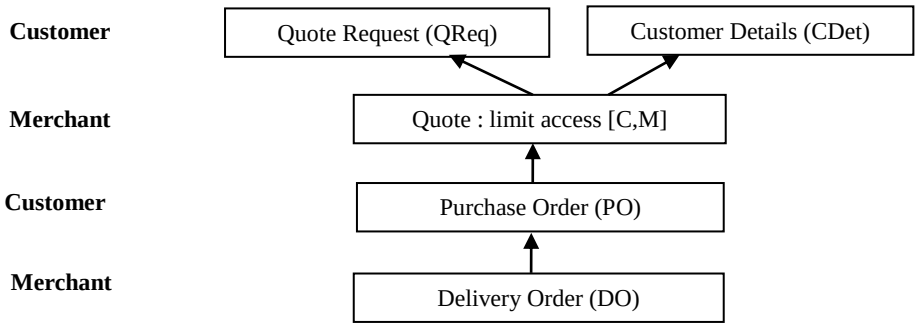


Figure 3.19 Dependencies between Knowledge Elements

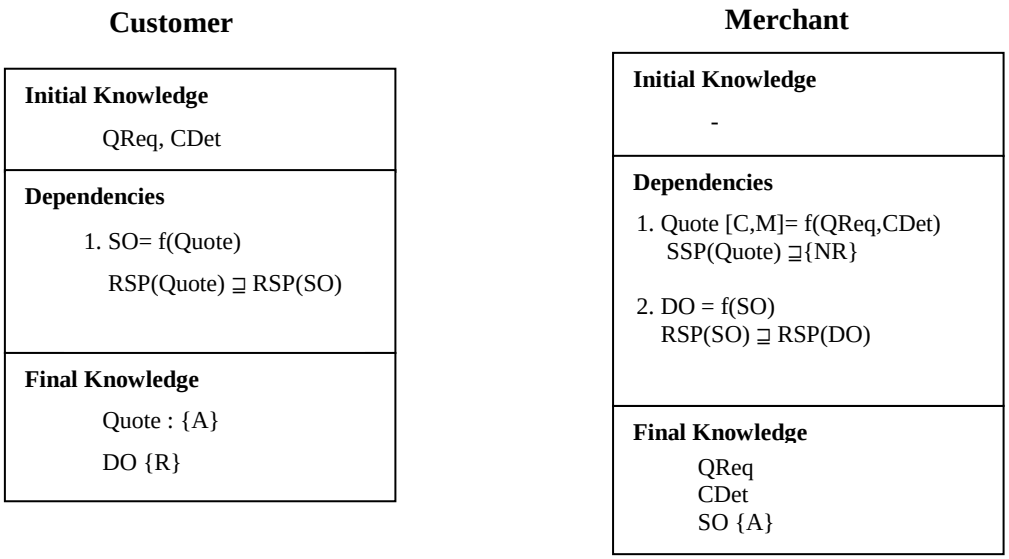


Figure 3.20 Initial and Final Requirements and Dependency Constraints

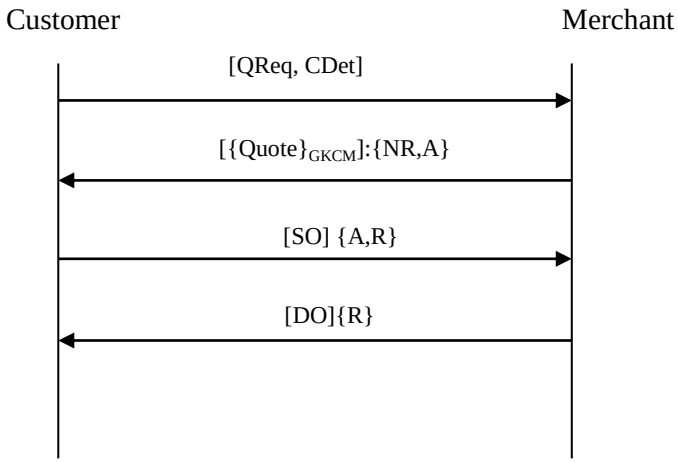


Figure 3.21 Protocol Specifications at Message Level

3.7.2 Performance of PGPS

The regeneration of the SET purchase protocol was used as the benchmark to measure the PGPS performance because it provides end-to-end security necessary for e-commerce and there have been past search based synthesis attempts to regenerate it. Unlike the earlier search based techniques, PGPS approach makes synthesis at runtime possible. A protocol similar to the SET purchase protocol was generated within 17 milliseconds using a 2.5 GHz laptop running a Windows XP operating systems. With faster protocol servers synthesis duration can be reduced significantly. Other search based synthesis techniques have relied mainly on abstract specifications of protocol goals using BAN logic (refer to Section 2.2.1.1). The performance of these techniques [23, 73] (with much longer reported duration for synthesis) cannot be compared directly with PGPS where proven schemes are combined to create the protocols.

3.8 Discussion

Past attempts to synthesise collaboration protocols between autonomous entities were mainly based on transitions and commitments permitted by their business rules. However, such attempts did not model the security assurances necessary to hold interacting entities accountable for their actions. Standard security protocols designed for e-commerce, such as SET, were limited to specific configurations. If protocols are to be synthesised with necessary security assurances, individual entities must be allowed to express the exact security requirements for all possible data exchanges. The lack of standard definitions for common security properties, such as authentication, led to many of the past protocol flaws. The PGPS framework makes it possible to specify different levels of assurances unambiguously by combining a few of the basic security properties. The fine-grained security properties formed by combining the basic properties make it possible to aggregate security requirements of interacting entities. To provide distinct schemes for all fine-grained security properties, PGPS uses a structured approach that combines challenge response mechanisms with compositional techniques. Furthermore PGPS schemes allow grouping and sequencing constraints necessary for e-commerce to be considered. PGPS also facilitate security performance trade-offs by expressing the bandwidth and computational cost of synthesised protocols in terms of the security strength of the underlying elements.

3.8.1 Security Requirements using Fine-Grained Security Levels

Hierarchical levels of common security properties were defined in the past to provide different levels of assurance [33]. Authentication based on injective agreement provides a stronger assurance than that based purely on aliveness as it includes evidence of a unique run. However, such a strong authentication property (with additional overheads) may not be necessary if the message was received in response to a recent message. Furthermore, if security needs of dynamically interacting entities are to be met, subtle security properties necessary must be unambiguously defined. For example, the PGPS composite security property combining recency with receiver non-repudiation provides the sender the evidence to prove the receipt of data despatched recently, in addition to assuring recipient of data recency. PGPS fine-grained security levels are defined combining basic security properties in a predefined order to standardize the meaning of such assurances. By using fine-grained security levels, security requirement along any message path can be expressed as a function of requirements by message originator and recipients, thus allowing end-to-end security requirements to be met. Though the fine-grained security properties defined are adequate to express different levels of security assurances defined in the past, they must be extended if newer properties such as anonymity are to be included.

3.8.2 Synthesising Protocols Using Proven Schemes

Recent attempts to synthesise protocols can be classified into three main categories. In the first category are challenge-response techniques, which were devised to provide basic security properties [21, 47]. In the second category are compositional techniques, which form more complex protocols from simpler ones [25, 26, 75, 78, 109]. In the third category, protocol goals stated in the form of beliefs were used to derive new protocols [23, 24, 60, 73]. Though these approaches were successful in discovering and deriving new protocols, the required processing made them unviable for synthesis at runtime. PGPS combines challenge-response schemes with composition techniques to arrive at provably secure schemes enforcing fine-grained properties. These schemes were combined to create realistic security protocols at runtime; a protocol designed to meet the SET purchase protocol requirements was created in 17 ms, which compares favourably with other search-based techniques. The correctness of all basic schemes was proved using SPCL, while the validity of composite schemes follows directly from the compositional approach used. The main limitation of the model is that it cannot resist every form of attack, though the additional terms incorporated make common attacks difficult to mount. Non-repudiation schemes used can only provide partial guarantees unless strong assumptions are made about communication channels with trust server and compliance of recipient. However, by modelling trust relationships (in Chapters 4 and 5) together with security schemes using a holistic approach the proposed framework allows trust relationships with noncompliant entities to be lowered or severed.

3.8.3 Extending Security Schemes for E-Commerce

A compositional approach to e-commerce security protocols requires devising schemes that can work in conjunction with business related constraints. Grouping constraints where two or more data elements must be despatched together are handled by aggregating their security requirements. Non-repudiation schemes were extended to include fair exchange features. A novel interleaving algorithm using piggy backing allows data elements to be despatched to multiple recipients in any specified order. The payload for piggy backed evidence is reduced by removing redundancies and combining elements such as nonces. Though the processing cost increases with the number of recipients, the elapsed time for protocol generation was found to be less than 20 ms even with 25 recipients and 5 different properties.

The validity of schemes devised for multiple recipients was proved using Strand Spaces. The low elapsed time makes it ideal for services that must be composed securely at runtime. The interleaving schemes, however, assume that all entities are honest and all protocol obligations will be discharged correctly. However, as this assumption may not always hold, the schemes devised in Chapter 5 makes all intermediaries accountable for their actions. Another limitation of the interleaved scheme is that it requires entities to lie along a straight path (in sequence), which increases the amount of piggybacking needed, especially for large numbers of recipients. Furthermore, the validity of interleaving schemes

depends on the trustworthiness of all the entities involved. The schemes devised in Chapter 4 select intermediaries based on category specific trust relationships and do not constrain them to lie along a straight path.

3.8.4 Cost Estimation Based on underlying Cryptographic elements

Security performance trade-offs have attracted much research interest in recent times. Many have argued that security protocols should be designed to achieve "good enough security" as security is about trade-offs rather than absolutes [45]. Security strength should be made to reflect the level of threat posed, with stronger keys used for protection against large organisations and governments, which have more resources at their disposal [85]. Private key operations are generally much slower than public key operations and their ratio grows linearly with key size. Performance issues become even more critical in wireless applications, where resource-limited devices are used. The computational constraints of such devices have led to severe performance degradation in the past due to high security overheads [52]. Energy usage is one of the main constraints for wireless devices as they are mostly battery operated. The energy requirement, too, is found to be highly sensitive to the size of asymmetric keys [110]. With widespread use of mobile applications needing security there is now a strong research interest in finding the right security/performance trade-offs [50-52, 92].

Past synthesis attempts modelling security at an abstract level did not consider the underlying protocol costs. PGPS makes security performance trade-offs possible by modelling security aspects together with computational costs and protocol bandwidth. The formal methods used in deriving individual and composite schemes ensure security goals are met. The cost model allows protocol bandwidth and computational cost to be expressed as a function of underlying security-strength, data security levels, number of entities and data-size. Such a model allows compatible services with least cost to be selected when using resource limited devices and networks. Furthermore, the security strength of protocols can be lowered if necessary, to meet performance/cost bottlenecks.

3.8.5 Future Work

The current PGPS model can be extended in a number of different directions. The security properties can be extended to include other recent properties in e-commerce such as anonymity and privacy. PGPS schemes can be further strengthened to resist other forms of attacks including reflection, oracle and algebraic attacks. Different schemes may be devised for security properties such as non-repudiation that provides the required level of trade-offs with performance. PGPS uses a compositional approach to generate security protocols. Specialized model checkers can be devised to verify whether the composed schemes continue to meet the original requirements. Currently composed properties use distinct encryption techniques combining signing and encryption. Such overheads for composite schemes can be reduced by devising techniques based on signcryption which combines signing and encryption into a single operation.

3.9 Conclusion

This chapter has presented a model that allows security protocols to be synthesised combining proven schemes. The issues of fine-grained security properties, basic and composite schemes, and trade-offs between security and cost have been addressed. In particular, this chapter presented the following solutions.

- Fine-grained security levels were formed combining six basic security properties that incorporated some features necessary for e-commerce (such as fair exchange). The meaning of each composite property was made precise by standardising the order in which individual properties should be enforced. Operators defined over fine-grained security properties allowed security levels along protocol paths to be set, aggregating end-to-end security requirements.

- The correctness of the challenge-response and public key-based schemes devised were proven using SPCL. Composite schemes were formed combining non-interfering basic schemes. The correctness of the composite schemes followed directly from the compositional logic used. Two-party schemes were strengthened to withstand common attacks by incorporating additional cryptographic elements.

- The two-party schemes were extended to multiple recipients using interleaving techniques. Such schemes meet the needs for specific applications such as contract signing, where data and evidence must be delivered in specified order. The correctness of these extended schemes was proved using Strand Spaces.

- The computational cost and bandwidth of synthesised protocols were estimated as a function of security strength and security levels used along the protocol path. Such an approach allowed the right security level to be chosen on the basis of performance constraints. Furthermore, when two or more protocols meet the security and functional requirements, computational costs and protocol bandwidth can be used as the basis for selecting the interaction with better security performance trade-offs.

Chapter 4: Promoting Trust via Endorsement Intermediaries

4.1 Introduction

Chapter 3 used cryptography for authenticating known entities and data originating from them. However, cryptography alone cannot secure e-commerce transactions between previously unknown entities [111]. Public key systems can only vouch for the association between an identity and a public key, and not for the trustworthiness of an entity [112]. Cryptography cannot prevent a dishonest merchant supplying faulty goods, even if the merchant's identity is known. Overcoming the perception of risk and insecurity in e-commerce therefore requires new trust promoting mechanisms [113]. Intermediary mechanisms help promote trust between unknown entities by combining trust and security [42, 53, 54]. Such mechanisms rely on economic incentives and disincentives to "weed out" malicious trading entities and intermediaries [38, 53]. Privacy considerations too may influence the trust placed on e-commerce intermediaries [114]. As trust plays a central role in e-commerce, trust between interacting entities must be considered a core requirement [115].

In traditional commerce, initial trust relationships established through reputation continue to evolve through direct experience. Reliable entities benefit through increased clientele and higher valued transactions. Similarly, reliable e-commerce entities can be made to benefit, if policies model trust relationships as a function of past conduct. Such policies must also reflect risk averseness of individual entities and domains, expressed through trust disposition and trusting beliefs. If the criteria for trust establishment and trust evolution are too stringent, business opportunities may be missed; if it is too relaxed, untrustworthy alliances may be formed. In general, false positives are less risky than false negatives as a missed trade through too stringent trust requirements is less harmful than trading with a dishonest entity through too relaxed trust requirements. Finding the right balance often requires statistical or simulation models as the risks and benefits vary significantly across categories and domains [119].

In the past, traditional commerce flourished only after institutional-trust promoting mechanisms were put in place in the form of guarantees, indemnities and endorsements. Endorsements by authorised intermediaries helped to promote trust between previously unknown entities. In a similar way, authorised e-commerce intermediaries can promote trust between unknown entities by endorsing the messages exchanged. For example, an e-commerce payment order from an unknown customer is more likely to be trusted by the merchant if it is endorsed by an authorised payment gateway. A news feed provider may want to ensure that all data published originated from reliable sources and is free from malicious contents by getting them endorsed. A games developer may want to send a new game to all subscribers endorsed for their trustworthiness. Indirect endorsements may be necessary if the message

originator and recipient do not share a common trusted intermediary. Intermediary frameworks must therefore allow both direct and indirect endorsements.

Promoting trust based on traditional commerce intermediaries poses a number of challenges. Intermediaries for traditional commerce are generally category specific. A JP (Justice of the Peace) may act as an intermediary endorsing a legal document but may not have any jurisdiction over financial matters. E-commerce intermediaries too should be classified according to the categories they are authorised to endorse. Traditional commerce provides legal recourse when intermediaries fail to discharge their obligations. Making e-commerce intermediaries accountable requires devising protocols that incorporate explicit evidence that can be presented to a third party. Furthermore, trust relationships between intermediaries in traditional systems are fine-grained (not binary), directly reflecting the extent of past transaction history. For example, *B* may be trusted as an intermediary between *A* and *C*, only if both *A* and *B*, as well as *B* and *C*, have very strong trust relationships. E-commerce intermediaries too can be selected in a similar way, if the extent of trust relationship is modelled using some quantifiable measure, such as the value of past transactions.

Whilst endorsement-intermediaries can help mitigate the risks inherent in e-commerce collaboration, they come at a price. Each additional intermediary increases endorsement costs, protocol bandwidth and response-time, and creates another avenue for security compromise. A large number of intermediaries may be inevitable in environments where only a few trust relationships exist or a long chain of hierarchical endorsements is necessary. In other environments, the cost of intermediaries may be considered too high in relation to the value of transactions. Hence, it is vital that a trust building framework for e-commerce provide the means for balancing risk mitigation with performance, cost and security, taking into consideration the type of environment and the value of transactions.

4.1.1 Literature Review

Trust forms the basis for e-commerce, web services and many evolving technologies such as cloud computing. Both direct experiences based on past interaction and recommendations by trusted parties impact how trust relationships evolve in such environments. Traditional commerce too has also relied on a number of instruments for building indirect trust, including services provided by authorised intermediaries such as bank guarantees, endorsements and credit checks [121-123]. Use of such authorised intermediaries has helped to mitigate the risks involved by making intermediaries liable for their specific endorsements or guarantees. If such a category specific intermediary architecture can be devised for e-commerce, the perception of risk and insecurity between trading entities can be reduced. To the best of our knowledge there has been no past attempt to model a category specific endorsement intermediary architecture. The following sections review related research involving trusted paths, the role of trust providers, trust propagation and transitive trust.

4.1.1.1 Trust Characteristics

Although there is no standard definition for trust in computer science [112], a number of trust characteristics have been generally agreed, including “trust is directed”, “trust is subjective” and “trust is category specific” [120]. The concept of trust in past research involves some level of uncertainty or risk for the trusting party [116]. This perception of risk is based on trusting beliefs about the ability, integrity and benevolence of the trustee [117]. Ability refers to the skills and competencies of the trustee in a specific domain; integrity concerns the moral and ethical standards of the trustee; benevolence describes the degree of goodwill the trustee has towards the trusting party [116, 118]. Other factors that influence trust are trust disposition and institutional trust. Trust disposition quantifies the extent to which an entity is willing to be influenced by other entities, while institutional trust is the confidence in the presence of well-defined structures that allow legal recourse [118]. If trust between unknown entities is to be promoted, the underlying beliefs and factors that facilitate trust should be measured and regulated. In recent work involving trustworthiness of service providers in cloud computing environment the need for fuzzy inference system that mimics the human mind has been proposed as a way to handle the uncertainty present in recommendations [161].

4.1.1.2 Trust Evolution

Trust evolution depends on direct experience and indirect referrals or recommendations. Though trust cannot be modelled precisely, past attempts define trust and trust growth as a function of transaction history and loyalty [139], which are similar to credit history which grows with positive experience over time. Similarly any negative experience causing poor transaction history such as a series of defaults in payments can adversely impact trust relationships. In e-commerce environments where trust relationships are dynamic the time frame and the price range for which trust values are applicable should be modelled explicitly using appropriate vectors [160]. Such a technique can prevent malicious agents building the trust through low valued transactions and attempting to cheat on a costly one. Others of recentness of past transaction when modelling trust; production rules lower trustworthiness more severely for recent poor conduct than those from distant past [162]. Others have devised self-organizing trust network using distributed algorithms to help safeguard against poor recommendations by malicious agents [163].

4.1.1.3 The Role of Trust Service Providers

Commercial entities are unlikely to take part in a transaction unless their own interests are safeguarded. For example, a person buying a product over the internet may not be willing to pay unless it is in good order and the seller may not be willing to send the product until payment has been received. Such stand-offs can be resolved by trust service providers (TSPs) which ensure commitments are made only after all parties have met their obligations [123-125]. However, specifying the exact role of a TSP for a specific interaction is not trivial. One past solution involves describing the acceptable and desirable states for all entities, before finding their intersection [37].

This scheme protects the interests of all entities by setting up an indemnity account, which can be forfeited when one of the entities fails to meet the condition. Explicit semantic constraints used in this method make verification of synthesised protocols possible. However, restricting interaction to two parties without any mechanisms for providing explicit evidence limits its applicability to a small class of exchange problems.

4.1.1.4 Economic Incentives for Trust Service Providers

Trust service providers play a specific role between two or more entities for a price. In one approach the economic incentive needed to keep TSPs reliable is modelled using a probabilistic trust model [38]. It is assumed that all TSPs are motivated by selfish behaviour for economic gain. The reliability of a TSP is equated to the probability that it can play the required role. Appropriate payment rules were devised to provide trusted agents with sufficient incentive to comply with trust assumptions. Payments necessary for each agent are such that their gains by complying with the trust requirements outweigh the gains of violating them. The novelty of this approach is that it models the economic benefit from the service provider's perspective and uses incentives to prevent breach of trust. The use of a fine-grained trust model allows a more realistic representation of trust. However, modelling trust based purely on short-term economic gain without considering the long-term benefits derived from strong trust relationships, limits its applicability. Furthermore, computing the payment necessary from individual entities requires collecting private information from agents (which may not be possible).

4.1.1.5 Need for Endorsements

Traditional commerce has relied on confidence building instruments, such as endorsements, to promote trade. Many government agencies provide endorsements for various services offered by private practitioners. E-commerce entities too can use such endorsement services to increase the level of confidence in services being offered [121]. Past proposals for e-commerce endorsements include the use of separate agencies for getting service credentials [121], and the use of endorsement proxies to select good services [126]. The level of credentials provided by endorsement agencies may vary depending on domain-knowledge. Confidence in such endorsements is inevitably dependent on the confidence in the endorser. An endorser licensed by a local, authorised organisation may have limited jurisdiction, confined only to that domain. If licensing agencies are to endorse services from other domains, the endorsements themselves must be endorsed by intermediaries at a higher level of jurisdiction. This is similar to insurance agencies being insured by other insurers. Although hierarchical transitive assurances are inevitable, long transitive chains of endorsements may lead to reduced confidence. Furthermore, endorsements based on static information are often not reliable. E-commerce and web services, however, require up-to-date information reflecting the trading entities' confidence in endorsement intermediaries and endorsement intermediaries' confidence in trading entities.

4.1.1.6 Trusted Paths

Dynamically determining e-commerce trusted paths relies on underlying mechanisms that model the evolution of trust relationships [127]. Many past algorithms combined multiple trust service providers (TSP) to build up the trust between interacting entities [42, 128, 129]. The metrics for TSPs can be based on positive past trust experience or private relationships. In one approach using a distributed algorithm, trusted paths are extended iteratively by exchanging information between entities until a path is found between message sender and recipient [128]. The TSPs themselves can be running in nodes in different environments, and executing a distributed search algorithm for building the trusted path. A forward message is sent to the adjacent node when searching for the target, and a backward message when a path is discovered. This approach, by using a parallel search procedure allows the search time to be reduced significantly. However, the number of messages grows exponentially with increasing node numbers due to each node forwarding the message to its neighbours, which results in congestion in highly trusted nodes. Furthermore, incorrect trust information by compromised entities may create invalid trusted paths.

4.1.1.7 Direct and Indirect Trust Propagation

Many web sites facilitate a form of trust propagation by displaying aggregate user scores. However, dishonest entities can distort the aggregate scores by placing biased views. Unscrupulous marketing agencies are known to pay others to write favourable feedback. Averting such malpractice requires restricting trust propagation to globally trusted information sources and other trusted entities. Empirical results have been used to compare the effectiveness of different means of direct and indirect propagation [130]. In direct propagation, if *X* trusts *Y* and *Y* trusts *Z*, then *X* ends up trusting *Z*. With propagation using co-citation, if *P* trusts *R* and *S* while *Q* trusts *R*, then *Q* also ends up trusting *S*. Propagation using transpose trust is applicable if *X* trusting *Y* causes *Y* to trust *X* to some extent. The propagation by co-citation was found to be the most effective predictor based on work done comparing different propagation methods [130]. Empirical study highlights the need to combine multiple strategies for trust propagation. However, the binary trust model where one entity either completely trusts or distrusts another, does not allow the extent of trust required to be modelled as a function of transaction value or risk.

4.1.1.8 Transitive Trust

Transitive trust refers to trust placed in an entity based on experience of others (not direct). Although transitive trust is an integral part of human communications, it is not easy to define precisely. Moreover, trust is not always transitive in real life [131]. For example, Alice's willingness to trust Bob to repay a loan for a specific amount and Bob's willing to do the same with Charlie, does not necessarily lead to Alice's willingness to do the same with Charlie. However, if Charlie trusts Dave to be a good plumber and Bob trusts Charlie to be able to recommend a good plumber, Bob may also trust Dave to be a good plumber. Similarly if Alice trusts Bob to recommend a plumber known to him

through referrals, Alice may be willing to trust Dave. Many trust management systems model trust by combining direct trust with transitive trust [120]. Traditional trading entities, too, have combined direct trust based on past experience with indirect trust relationships to increase their trading opportunities. Though indirect trust increases trading opportunities they also open up more venues for attacks [122].

Public key infrastructure (PKI), which uses a form of transitive trust aggregates functional and referral trust [54, 132]. An entity trusting a certification authority (CA) must have the functional trust that the public key of CA is authentic and correctly certifies public keys, and a referral trust that the CA validates other subordinate CAs able to certify public keys. This process is iteratively repeated until the public key of a source is obtained. Thus, the last edge in a trusted path represents **functional trust** while all other edges represent **referral trust** [54]. Use of distinct edges for functional and referral trusts in transitive trust graphs allows multiple referrals through parallel trust combinations to be represented [54]. However, the parallel combination of trusted paths used results in complex cyclic networks that can only be analysed using heuristic techniques. Moreover, the trusted paths are created without considering the value and category.

Transitive trust plays a major role in service oriented social networks which are becoming widespread. Trustworthiness of service provider must be expressed in terms of transitive trust along different social trust paths. Computing trustworthiness however, requires heuristic algorithms as the optimal path algorithms are NP-Complete [133]. Social network communities are also less amenable to a hierarchical organization based on the type of communities as some nodes may belong to multiple communities [134]. Therefore recent work has treated communities as nodes instead of links [135]. Trustworthiness however, cannot be expressed in terms of existing trust relationships alone as trust degrades with the number of intermediaries. In social networks where most pairs can be connected using at most six trusted intermediaries [133, 136], attenuation in transitive trust is modelled implicitly by setting it to the product of all trust relationships (that lie in the range 0 to 1).

Trust also plays a dominant role in service oriented computing (SOC), a software architecture that allows independent services to be composed in real-time. In SOC, services that are both autonomous and persistent collaborate together to perform the required actions. Trust plays a major role, as the service user has no direct access to secondary service providers operating under different sets of administrative structures. Prior dealings and referrals from other trusted entities can be used in selecting services. To ensure services do not work against the interests of service users, services must be selected based on compatible security and privacy policies [4, 16]. The underlying framework therefore must allow trust and security issues to be modelled together.

Referral networks help identify service partners by maintaining and sharing the trustworthiness of other services [125]. Such a mechanism fosters good behaviour in trusted entities, as a single breach of trust may damage trust built up over a long period. An agent needing a specific service contacts its neighbours, which may offer the service or give referrals to other agents. The agent requesting the service may accept the service or follow up with the referrals. Services are matched based on the quality of service provided and the level of service requested. When a service is selected and used, its ratings and referrals that led to that service are updated on the basis of the quality of service. These updates lead to constant evolution of these trust networks, whereby agents move closer to the ones that they trust. The main benefit of this approach is that principals help each other discover trustworthy services. This approach is also less risky than the traditional approach, where there is sole reliance on the integrity of some central authority. However, these networks may not be effective when the number of referrals is too high, or too low; higher cost is incurred through increased service options for high number of referrals while no matches may be found for a low number of referrals.

4.1.1.9 Summary of Literature Review and Evaluation

Although no standard definition exists, trust is assumed to involve some level of uncertainty or risk. Trust is considered to be directed, category specific and subjective. Factors that influence trust include trust disposition, institutional trust and trusting beliefs. Initial trust is often established through some form of trust propagation mechanism. Trust relationships continue to grow over time on the basis of past experience. Trust promoting mechanisms include trusted intermediaries (trust service providers), with varying roles depending on the type of interaction. Sufficient economic incentives disincentives must be provided to prevent intermediaries breaching trust assumptions. Endorsing transactions by authorised endorsement intermediaries may help increase the confidence between unknown entities.

An extensive literature survey revealed the need for a framework that combines trust evolution and trust transfer with institutional trust using some form of endorsements [113, 126]. Such endorsements must be both hierarchical and category specific as endorsements in e-commerce are category specific and may be subject to endorsement at another level. Unlike traditional e-commerce intermediaries, category specific endorsement requires both direct and indirect endorsements. For example, a liquor purchase message from a retailer entity to a wholesale merchant entity can be routed through a liquor purchase endorser, one or more liquor endorsement intermediaries and a liquor sales endorser. Thus, direct and indirect endorsers play a role similar to that of functional and referral trust in transitive trust systems. If trusted intermediaries are to be made reliable, mechanisms must be devised to detect trust breaches. Proven conduct should be used as the basis for growth or decay in trust relationships and extension or termination of endorsement capabilities. Another gap identified is the transition between transitive and direct trust. In traditional commerce transitive trust often evolves into direct trust which carries with it a higher level of reward and risk. Although much work has been done with direct and transitive trust, little or no work has been done modelling the transition from transitive to direct trust.

To address these gaps this chapter presents a model named Promoting Trust in Endorsement Intermediaries (PTEI), based on a hierarchical institutional framework, trust disposition and trusting beliefs. An institutional framework is provided through category specific endorsement intermediaries. The trust disposition and trusting beliefs of individual entities and domains form the basis of trust transfer and trust evolution policies. PTEI policies are designed to promote trust relationships with reliable trading entities and intermediaries through rewards for compliance and penalties for trust breach. Increased trust coupling and strong trust relationships create more trading opportunities for entities and greater endorsement incomes for intermediaries. Trust transfer allows transitive depths to be reduced eventually leading to direct trust relationships. Although direct trust carries greater rewards the trust transfer threshold should be set experimentally to reduce risks resulting from poor alliances.

4.1.2 Main Contribution

The overall contribution of this chapter is an institutional framework named PTEI for promoting trust in e-commerce [167]. It presented a number of solutions which are outlined below.

- The first contribution is an architecture that helps to promote trust in e-commerce by endorsing all key data exchanged between unknown entities using authorised category specific intermediaries. The resulting trust based on both past experience and the authority granted to endorsers at different levels, provides a much stronger basis for trust propagation than traditional transitive trust models. Hence, the endorsement based trust proposed degrades at a much lower rate (by varying the associated parameters) than traditional transitive trust. Furthermore, by combining endorsements and category specific trust this model makes e-commerce transactions without human involvement possible for more critical transactions such as those involving medical drugs.

- The second contribution is a self-regulating centralized trust network which allows evolution of existing trust relationships and formation of new trust relationships based on past conduct, trust disposition and trusting beliefs. Simulation results for the model show that the number and extent of trust relationships is proportional to the reliability of entities. Such a model corroborates well with traditional commerce where the number of alliances with traders and agents grows at a rate proportional to their trustworthiness.

- The third contribution is the approach used to allow trusted path generation at runtime. Centralized network used for modelling trust relationships makes such path generation possible. For large networks, PTEI groups entities into hierarchical domains on the basis of trust coupling. The simulation results and the complexity analysis done for the hierarchical network, suggests that trusted paths through a network of 800,000 nodes can be created within one second. The retrieval times can be further reduced by using the domain trust trees formed offline, which maintain separate trusted paths from each entity.

4.2 Statement of the Problem

Endorsements by category specific intermediaries form the key to promoting institutional trust in traditional commerce. Using such endorsements in e-commerce poses a number of problems: modelling category specific hierarchical endorsements, techniques for evolving trust relationships and algorithms for finding trusted paths in at runtime. These are highlighted in this section.

- Traditional commerce and law enforcement relies on the presence of category specific endorsement authorities. For example, a person may not teach in a primary school until that person's character is endorsed by police authorities who have access to criminal records.
- In traditional commerce, trust relationships evolve dynamically based on past direct experience while new trust relationships are formed based on cumulative experience of trusted parties. If such dynamic trust relationships are to be modelled for e-commerce, new trust transfer and trust evolution mechanisms must be devised.
- Trusted endorsement paths may be created based on a number of factors, including minimum trust relationships along the path, transitive depth, cost of endorsements, or a combination of these. For example, longer endorsement paths may be acceptable in sensor networks used in weather data but not in wired networks transmitting financial transactions.
- Dynamic collaboration between unknown e-commerce entities depends on finding endorsement paths through intermediaries at runtime. However, generating such paths for large networks can become intractable even if algorithms are of polynomial complexity.

4.2.1 Research Questions

Before an institutional trust framework similar to that found in traditional commerce can be created, the following research questions must be answered:

- Endorsements formed the basis for institutional trust in traditional commerce. This leads to the research question: How can an institutional trust framework for e-commerce be designed which allows vital data to be endorsed by authorised category specific intermediaries?
- If trusted paths are to be generated dynamically, trust relationships must be maintained centrally. In traditional commerce, existing trust relationships continue to evolve through direct experience while new trust relationships are established through recommendations. This leads to the research question: How can a centralized trust network allow establishment and evolution of trust relationships?
- Trusted paths for e-commerce may require data to be passed through large networks involving many hierarchical domains. This leads to the research question: How can the algorithms and data structures used for generating trusted path be made scalable?

4.3 Outline of the Solution

Endorsement trust has played a vital role in traditional commerce. Consider a company that only hires an electrician endorsed by a maintenance service agency it trusts when some electrical work needs to be done. Such endorsements are category specific; a maintenance agency, for example, cannot provide product endorsement. Endorsement trust improves gradually based on positive experiences, while it may decline sharply if there is one single negative experience. A bank, for example, is more likely to guarantee a large line of credit to a customer with a long and positive transaction history than to a customer with a short one. However, one bad experience may seriously damage the trust capital built up, and affect all future guarantees. Endorsement trust in traditional businesses reflects the beliefs of both the endorser and the endorsee. For example, the endorsement trust between an agency and a particular electrician reflects both the agency's beliefs based on past electrical work done and the electrician's beliefs based on the type of companies hiring through the agent.

Trust relationships in general are treated as one-way functions, as B may trust C's recommendation of good stock picks but not vice-versa. If an e-commerce framework is to model trading opportunities based on traditional commerce, the underlying trust relationships must be allowed to evolve over time. Lack of frameworks that allow reasoning about trust based on users' trust policies has prevented e-commerce from exploiting many new opportunities [137, 138]. Thus, the proposed endorsement trust based architecture incorporates the following:

- non-binary trust relationships that reflect the level of confidence
- category specific intermediaries able to provide the necessary endorsements
- a hierarchical structure for endorsement intermediaries
- a trust network maintaining the trust relationships between intermediaries and trading entities
- trust evolution based on past experience
- formation of new trust relationships reflecting reputation and trust disposition.

TERMINOLOGY 4.1 COMMON TERMS

The terminology used for common terms in this chapter is described below:

- ***Entity*** *(EN) represents either a trading entity or intermediary.*
- ***Category*** *(C) of data (d) is the aggregation of categories of data elements as in:* $C(d=d_1 \cup d_{2...} \cup d_n) = C(d_1) \cup C(d_2) \ldots C(d_n)$ *where* $d_1, d_2..d_n$ *are atomic data elements.*
- ***Endorsement Capability*** *of an intermediary is the set of knowledge element categories it is authorised to endorse.*
- ***Trust Disposition*** *is the extent to which an entity has a tendency to be willing to depend on others [113]. In this thesis trust disposition is considered to be category specific.*
- ***Direct Endorsements*** *are performed based on direct knowledge of entities in the domain while* ***Indirect Endorsements*** *are performed based on trust placed on other endorsers.*

- ***Domains are*** *groupings of trading entities and intermediaries. These grouping may be based on geography or other factors such as existing trust relationships.*
- ***Transitive Depth*** *is the number of intermediaries between the data originator and recipient.*
- ***Maximum Transitive Depth*** *is the maximum number of intermediaries allowed in a given domain. This value is set as a function of the number of entities and the number trust relationships in that domain as described in Section 4.4.9.*
- ***Entity and Intermediary Reliability*** *denotes the average number of transactions before a transaction failure occurs in a given domain and category. Invalid endorsements and actions by intermediaries can be detected using the accountability schemes presented in Chapter 5.*
- ***Trust transfer*** *establishes new direct trust relationships based on transitive trust.*
- ***Trust evolution*** *is the change in endorsement trust relationship based on past experience. The extent of change varies across domains and categories.*

The proposed framework helps to mitigate risks involved in trading with unknown entities by routing key data through hierarchical category specific endorsement intermediaries. Intermediaries may perform direct endorsements based on direct knowledge of trading entities or indirect endorsements where endorsements themselves are endorsed. Both direct and indirect endorsers are selected on the basis of existing trust relationships which are maintained by a centralised trust server. Trust relationships improve with positive experiences but decline when invalid endorsements or trading entity failures are detected. Selection criteria for trusted paths can be expressed in terms of transitive depth, cost of endorsements or trust relationships along the path. Path selection for large trust networks is made efficient by subdividing them into hierarchical trust domains.

- Intermediaries in PTEI are classified into direct intermediaries which base their endorsements on direct domain knowledge of entities, and indirect intermediaries, which endorse endorsements by other trusted intermediaries (such as companies insuring other insurers). Indirect intermediaries are organised into multiple levels to facilitate a hierarchical structure that reflects the extent of their jurisdiction.
- Entity level policies reflecting trust disposition determine the trust transfer threshold and the duration of distrust with entities breaching trust. Group level policies specify trust evolution rate and maximum transitive depth for all entities in the domain.
- Objective functions with varying weights for different factors allow flexible path selection. Inverse values are used for factors to be minimised, such as cost and depth.
- PTEI trust networks are decomposed into domains on the basis of physical or logical boundaries. Separate domain trust sub-trees are created for different objective criteria. The root node of each such sub-tree is labelled with all entities that can be reached as well as their trust and performance measurements. By clustering entities into hierarchical domains trusted path retrieval for large networks can be distributed to individual domains that can work independently.

4.4 Endorsement-Trust Model and Building Blocks

The proposed architecture, PTEI, can help promote confidence between unknown e-commerce entities by allowing key messages to be sent through category specific endorsement intermediaries. An endorsement intermediary may perform direct endorsements based on its knowledge of trading entities in its domain or indirect endorsements based on trust placed on other endorsement intermediaries. Direct endorsers use information from other services in that domain such as licensing for specific category, creditworthiness, legal status etc. Indirect endorsers may be organized into hierarchical levels such as provincial, national etc. Therefore the service provided by the endorser may vary based on the category, the domain, the hierarchical level and the type of endorsement (direct/indirect). Together endorsements provide a form of indemnity for trading entities.

Figure 4.1 shows the main components and functionality of the proposed architecture PTEI. The PTEI components allow maintenance of trust relationships, policies for various categories and domains, and tracking of endorsement processes. Trusted paths through intermediaries are formed on the basis of data category, source, recipients and trust parameters. Endorsement costs are reduced by sending data to multiple recipients through common intermediaries. Trust criteria specified are maximised subject to maximum transitive depth and minimum trust relationship along the path not being violated. These criteria are set reflecting the value of data, the category and the endorsement costs. Trust values are lowered (or severed) when noncompliance is detected during a transaction based on invalid evidence presented or afterwards based on archived evidence. Any intermediary endorsement found to be invalid may cause its endorsement capability to be revoked. Trust relationships improve over time when transactions are completed successfully. These economic incentive schemes are designed to cause growth in trust relationships between reliable trading entities and intermediaries, which in turn increases future trading opportunities and incomes. Intermediaries and entities are made accountable for their actions using end-to-end cryptographic schemes, presented in Chapter 5. These schemes require endorsers to maintain any evidence of information used as the basis for their own endorsements.

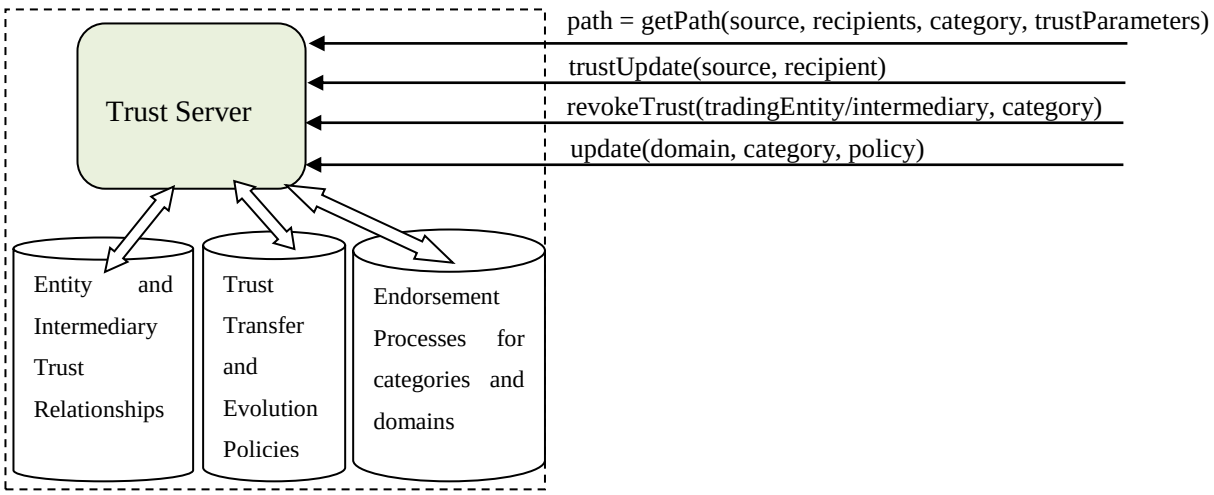


Figure 4.1. PTEI Model Overview

PTEI maintains trust capital in a hierarchical category specific trust network. The main functions of PTEI can be can be classified based on when they are carried out:

Immediately Before Transaction:

1. Trusted path through endorsement intermediaries is selected based on trust parameters
2. Selected path is certified by Trust Server

For Trust breaches detected during Transactions:

3. Devalue or sever trust relationships along the trusted path based on domain policies; seek alternate paths if necessary.

For Transactions Successfully Completed:

4. Update trust values based on domain policies

Detect Problems (manually) within a Specific Period after Transaction:

5. If a problem detected after the completion of transaction is attributed to an intermediary endorsement failure, trust relationships along the path are devalued or severed. In addition endorsement capability may be revoked for a period based on domain policies. (Refer to Section 4.4.10.3)

The rest of this section presents the building blocks used in PTEI. Sections 4.4.1 and 4.4.2 define category specific endorsement intermediaries and endorsement-trust relationships. Section 4.4.3 presents category specific endorsement trust network. Section 4.4.4 outlines endorsement processes. Section 4.4.5 presents entity and domain policies that make the trust network self-regulating. Section 4.4.6 defines path trust and path endorsement trust. Sections 4.4.7 and 4.4.8 describe trusted path representation and selection criteria used in PTEI. Section 4.4.9 expresses maximum transitive depth required as a function of trust coupling. Section 4.4.10 describes trust and trust evolution policies.

4.4.1 Hierarchical Category Specific Endorsement Intermediaries

Traditional commerce has relied on some form of category specific endorsements for exchanging key information. A school admission procedure usually requires a child's details to be supported by a birth certificate (a document endorsed by a birth registrar). When multiple endorsements are required, a higher level of endorsement may not be granted until a lower level one is obtained. Police clearance at national level may be subject to state level police clearance. PTEI uses a similar structure by supporting category specific intermediaries. Intermediaries are classified according to the category of knowledge elements they can endorse. A message may contain knowledge elements belonging to one or more categories, labelled C_0, C_1 … C_n. Those in category C_0 need no endorsements, and can either be sent directly between trading entities or sent together with knowledge elements of any other category.

PTEI intermediaries can act as direct endorsers, indirect endorsers or both; direct ones endorse trading entities despatching and receiving messages, while indirect ones endorse other intermediaries. Direct

endorsers must have the necessary domain knowledge to perform the required endorsements. For example, any credit agency directly endorsing a loan request from a customer must have the authority and the means to access the credit histories of all customers in the domain. If the same credit agency were to endorse the loan request to the recipient bank as well, it also would have to possess the authority and the means to access the credentials of all banks. When no common intermediary exists, a path through indirect endorsement intermediaries must be found.

E-commerce trading between unknown entities from different domains poses many challenges. For example, regulations affecting any form of trading in restricted items such as arms and endangered animals may be imposed at many levels, including provincial, national and global. Sanctions at global level may override trade treaties at national or state level. In the proposed framework, authorised endorsement intermediaries are organized into hierarchical levels to enforce such regulations. At the lowest level, direct endorsers verify that such trading can be carried out by the trading entities based on access to various records in those domains. Indirect endorsements are performed, subject to meeting endorsements at other levels and meeting any regulations specific to that level in that category. PTEI selects such intermediaries on the basis of the level at which they are authorized to endorse and their existing trust relationships with other intermediaries. Such a hierarchical organization however depends on developing schemes that make them accountable for their actions.

4.4.2 Endorsement-Trust Relationships

Traditional commerce relied on agents and brokers to promote trust between unknown entities. Web based commerce has now replaced such human intermediaries and their associated costs by using specialised intermediaries such as Ebay, which maintain trust relationships based on past trading experience. In PTEI, an endorsement-trust relationship reflects an endorser's confidence based on past positive experience, thus serving as a predictor of future success. Endorsement trust relationships have three categories, depending on the types of end nodes as shown in Figure 4.2. The endorsement trust relationship directed from a trading entity to an endorsement intermediary reflects the endorser's confidence that the trading entity is valid for the specific data category. For example, if a "supply drugs" message from entity *A* is to be endorsed by a medical agent, the agent must believe that entity *A* is a valid originator for that category based on past experience. An endorsement trust relationship directed from an intermediary to a trading entity reflects the intermediary's confidence in the trading entity being valid for the specific data category. The relationship directed between two endorsers reflects the mutual confidence of one endorser on the other.

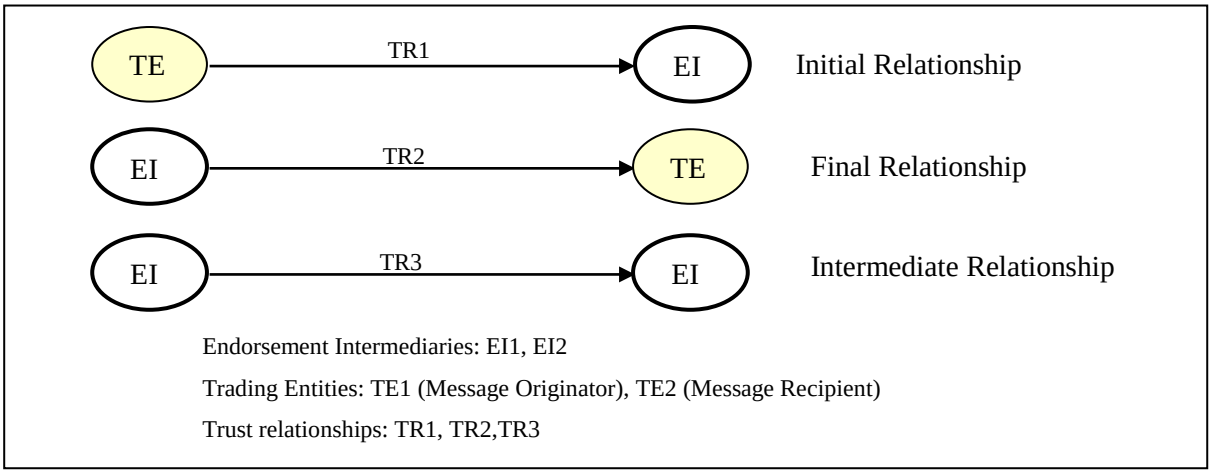


Figure 4.2 Types of Endorsement Relationships

Endorsement-trust relationships are directed, category specific and are assumed to grow gradually using non-binary values. When a trust relationship is first established it starts with the initial value of *TRMIN*, and grows through trust evolution and trust transfer (Section 4.4.10) up to a maximum value of *TRMAX*. Trust relationships are allowed to grow at different rates for different categories and domains. If two entities are not known to each other, or any prior relationship has been terminated because of trust breach, the value for the trust relationship is set to 0. The trust relationship with any trading entity (originator or recipient) may be terminated by direct endorsers performing a category specific search in the domain (such as a bankruptcy register). The trust relationship with an endorsement intermediary may be terminated at a later time, if direct endorsements are found to be invalid or when an indirect endorsement has been performed without any valid basis (without evidence of prior valid endorsement). The accountability schemes presented in Section 5.6 include explicit cryptographic evidences to detect and prove trust violation by endorsement intermediaries.

DEFINITION 4.1 ENDORSEMENT TRUST RELATIONSHIP

Let EI be the set of all Endorsement Intermediaries, TE be the set of all trading entities, C be the set of all Categories and TRMAX be a predefined scalar representing the maximum trust value

Then the Endorsement Trust Relationship TR *is defined as follows:*

- *When the sender is a trading entitiy and the recipient is an endorsement intermediary,* TR *represents the initial trust relationship as in :* $TR: TE \times EI \times C \rightarrow [0 - TR_{MAX}]$
- *When both the sender and the recipient are both endorsement intermediaries,* TR *represents the intermediary trust relationship as in :* $TR: EI \times EI \times C \rightarrow [0 - TR_{MAX}]$
- *When the sender is an endorsement intermediary and the recipient is a trading entitiy,* TR *represents the final trust relationship as in :* $TR: EI \times TE \times C \rightarrow [0 - TR_{MAX}]$

Note, no direct trust relationship exists betweeen trading entities ($TE \times TE \times C$).

Trust coupling (TC) is the average number of trust relationships (TR) that exist in a given domain, that is, the total number of trust relationships divided by the number of entities in that domain. Trust coupling is considered low if it is less than or equal to two.

DEFINITION 4.2 TRUST COUPLING

$$TC = \frac{\sum_{i=1}^{n} \#(TRi)}{n}$$

where $\#(TR_i)$ is the number of trust relationships in the i^{th} entity and n is the domain size.

4.4.3 Hierarchical Category Specific Intermediary Network

PTEI maintains a category specific intermediary network using a multigraph, which allows multiple edges between entities. Each entity in the network represents either a trading entity or an endorsement intermediary. This section outlines some of the design considerations of the PTEI model and some features of the underlying network before presenting a more formal model of the network.

Design Considerations	Trust Network features
An endorser should be allowed to endorse one or more categories from the set $C \setminus \{C_0\}$ where C_0 is the category that requires no endorsement	Multiple endorsement capability allowed. Separate edges used for representing trust relationships for different categories.
Trust relationship along the data path should exceed the minimum trust relationship specified for that data category.	Edges representing category specific trust relationships store the existing level of trust.
All categories other than C_0 must be exchanged through category specific endorsement intermediaries	No direct links (edges) exist between trading entities for categories other than C_0. Links for C_0 are not maintained explicitly.
Trading entities should be allowed to despatch data elements belonging to different categories grouped together, allowing intermediaries to verify they meet the necessary condition for endorsement.	Data elements can be sent together between trading entities subject to intermediaries having combined endorsement capabilities and edges for combined categories.
Hierarchical structure may or may not be imposed.	If imposed, inherent constraints will prevent direct links being formed between entities at incompatible levels (say level 2 and 4)

Endorsement capability of an intermediary (*ECAP)* is the nonempty set consisting of all categories other than C_0 it is authorised to endorse.

DEFINITION 4.3 ENDORSEMENT CAPABILITY (ECAP)

Endorsement capability of an intermediary ECAP is a non-empty subset of $\{C_1,C_2 \ldots C_N\}$ *where* $C_1,C_2 \ldots C_N$ *are the categories that require endorsements.*

Note *ECAP* cannot be empty and does not include C_0, the category that requires no endorsement.

DEFINITION 4.4 TRUST NETWORK (TN)

Let EN be the set of all entities and

ED be the set of all edges representing category specific trust relationships,

then Trust Network TN(EN,ED) is a directed weighted multigraph over EN and ED.

- *EN is either a trading entity TE, or an endorsement intermediary EI.*
- *TE is a tuple over <D,TEL> where D is the set of domains and TEL is the set of labels for trading entities.*
- *EI is a tuple over <D,EIL, ECAP> where where D is the set of domains, EIL is the set of labels endorsement intermediaries and ECAP is the endorsement capability.*
- *Each edge e in ED is labelled with category specific trust label of the* $<C_i,v>$ *where* $C_i \in C\backslash\{C_0\}$ and $v \in$ [*TRMIN,TRMAX*], the extent of trust relationship.

An Example using the Trust Network Multigraph

Figure 4.3 shows a network made up of eight intermediaries, I_1 to I_8, and six trading entities, *A* to *F* from four different hierarchical domains. Each edge in the network represents the extent of a trust relationship for a specific category using two separate labels, one for category and one for the extent of trust relationship. All data exchanged between trading entities other than that of category C_0 are assumed to require endorsements by intermediaries. Each intermediary node is labelled with its endorsement capability. In Figure 4.3 the intermediary I_1 $(\{C_1,C_2,C_3,C_4\})$ has greater endorsement capability than I_2 $(\{C_1,C_2,C_3\})$. Data can be sent endorsed between two trading entities along a specific path only if intermediaries exist with endorsement capabilities exceeding (or equalling) the data category and data category specific edges between them with the required trust level. For example, data of category $\{C_1\}$ can be sent from *B* in domain D_1 to *D* in domain D_3 via $B \rightarrow I_1 \rightarrow I_2 \rightarrow I_4 \rightarrow I_8 \rightarrow D$ provided the trust requirement for data does not exceed 2, the lowest category specific (C_1) trust relationship along that path. Note any other data element of type $\{C_0\}$ (which may provide the necessary context for data) can be sent together along the same path as they require no endorsements.

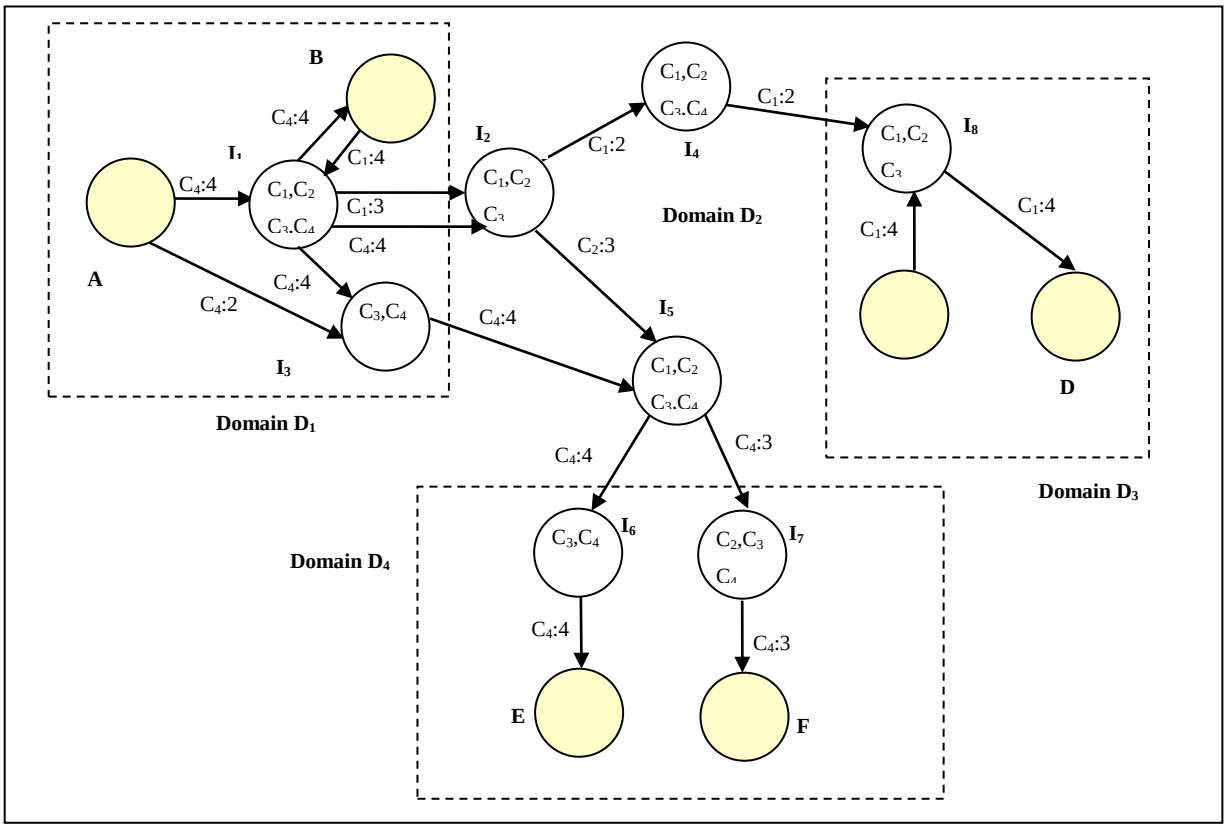


Figure 4.3 Network Showing Endorsement Paths and Category Specific Trust Relationships

The entity *A* from domain *D1* sending a message containing *{C4}* to *E* in domain *D4* may either send it through $I_1 \rightarrow I_3 \rightarrow I_5 \rightarrow I_6$ or through intermediaries $I_3 \rightarrow I_5 \rightarrow I_6$. The first route provides a better minimum trust relationship along the path (4) while the second route provides a better transitive depth (3). In general, the intermediary paths reflect the selection criteria used, such as transitive depth and maximum path trust, which is described in detail in the next section.

4.4.4 Endorsement Processes

Endorsement processes are services used by endorsement intermediaries. The process carried out by any specific intermediary can be uniquely determined based on the category and the type of endorsement (initial, final or intermediary).

DEFINITION 4.5 ENDORSEMENT PROCESS (EP)

An Endorsement Process EP = <EIL,C,ET,F> is a tuple over the set of all endorsement intermediary labels EIL, set of all categories C, endorsement types ET = {Initial,Final,Indirect} and the set of endorsement function F which carries out the endorsements. The endorsement functions specify exact output based on domain and category based rules, data and previous endorsement if any.

These functions are used as part of the security schemes presented in Chapter 5. Implementation details of intermediary endorsement functions are left open as they are category and domain dependent. For example, a notary in Myanmar may carry out different actions from an auditor in Canada.

4.4.5 Entity and Domain Trust Policies

PTEI policies are designed to promote trust based on both past direct experience [139,160] and indirect experience based on recommendations of trusted parties [40, 61, 128, 133]. Trust disposition and trusting beliefs are modelled using entity and domain policies respectively. Entity trust policies determine when new relationships can be formed reflecting the trust disposition of individual entity. Domain trust policies specify the rate at which trust relationships grow for positive conduct and decline for proven trust breaches. Domain policies also specify the maximum transitive depth allowed in terms of the number of entities. These rates aggregate trusting beliefs of entities in the domain. Both policies specify different rates for different categories reflecting their level of risks and rewards.

Entity Trust Policy specifies trust disposition for each category an entity is allowed to endorse in terms of trust transfer threshold. Trust transfer threshold (*TTT*) is the minimum reputation necessary for forming new trust relationships.

DEFINITION 4.6 ENTITY TRUST POLICY (ETP)

Let EN be the set of all entities,
C be the set of all categories and
TTT a natural number representing the trust transfer threshold.

Then Entity Trust Policy ETP = <EN,C,TTT > specifies category C specific trust disposition for each entity EN in terms of trusts transfer threshold TTT.

Domain trust policy specifies the rate at which trust relationships grow or decline for positive or negative conduct, the maximum number of intermediaries allowed and the period for which any entity or intermediary trust relationships is severed for trust breach or endorsement failure. Distinct values are maintained for different categories as PTEI network maintains category specific trust relationships.

DEFINITION 4.7 DOMAIN TRUST POLICY (DTP)

Let DL be the set of all domain labels,
C be the set of categories,
TER $\in$ [0,1] representing trust evolution rate,
TDR $\in$ [0,1] representing trust degradation rate,
MTD a natural number representing the maximum number of intermediaries allowed,
TSP: N trust severance period.

The domain trust policy DTP = <DL,C, TER,TDR, MTD,TSP> specifies category C specific trust values for domain DL including TER the rate at which trust relationships grow for valid behaviour, TDR the rate at which trust relationships degrade for poor conduct, MTD the maximum number of intermediaries in the domain and TSP the period for which trust relationships may be severed.

4.4.6 Path Trust (PT) and Path Endorsement Trust (PET)

Trust relationships in transitive networks are commonly modelled using the product of all trust relationships (in the range [0,1]) along the message path [129, 139]. Such a model reflects the intuition that transitive trust diminishes significantly with each additional intermediary. Path endorsement trust proposed degrades at a much lower rate as any trust violation by authorised intermediaries leads to much greater penalties; loss of endorsement capability as well as trust capital.

Path Trust (PT) for data d sent through a given path T is the lowest trust relationship along the path for any of the knowledge categories of its base elements.

DEFINITION 4.8 PATH TRUST

Let d be the data with data elements $\{d_j .. d_m\}$ *and*

$T = <E_0,E_1,E_2 \ldots E_n>$ *be the trusted path from origin* E_0 *to recipient* E_n.

Then the Path Trust for data d sent from origin E_0 *to* E_n *is:*

$$PT(d,E_n,T) = \min_{i=0}^{n-1} \min_{j=1}^{j=m} TR(KEC(d_j),E_i,E_{i+1})$$

Example:

Table 4.1 shows an example of existing trust relationships between entities *A, B* and intermediaries I_1, I_2 and I_3 for knowledge category $\{C_1\}$. Figure 4.4 shows two different trusted paths from *A* to *B* for data d with category $\{C_1\}$. The path trusts for $A \rightarrow I_1 \rightarrow I_2 \rightarrow B$ *and* $A \rightarrow I_3 \rightarrow B$ are 4 and 2 respectively based on the trust relationships along the paths shown. The first path chosen will reflect whether path trust or transitive depth is used as the criteria (Terminology in Section 4.3).

Entities	Extent of Trust relationship
$A \rightarrow I_1$	C_1:4
$I1 \rightarrow I_2$	C_1:5
$I2 \rightarrow B$	C_1:4
$A \rightarrow I_3$	C_1:2
$I_3 \rightarrow B$	C_1:4

Table 4.1 Extent of Trust Relationship (Category C1)

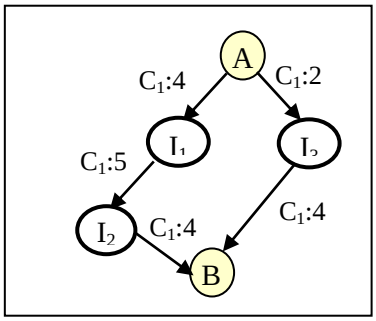


Figure 4.4 Trusted Paths using Different Criteria

Though path trust gives a measure of indirect trust between two entities, it does not model the trust attenuation likely to happen with each additional intermediary. Furthermore, it does not give an intuitive measure of indirect trust, as it can range from 0 to TR_{MAX} (a variable). Therefore, Path Endorsement Trust *(PET)* is defined to give an estimate of indemnity provided by the trusted endorsement intermediaries that lies in the range 0 to 1.

DEFINITION 4.9 PATH ENDORSEMENT TRUST

Let *PT be the Path Trust for data d sent to E_n along the path T,*

TD(T) be the transitive depth of path T

MTD be the maximum transitive depth for the domain

TR_{MAX} be the maximum trust relationship

Then Path Endorsement Trust PET (d,E_n,T) = $\frac{PT(d,E_n,T)}{TR_{MAX}}(1-\frac{(TD(T)-1)}{MTD})$ *gives a measure of indemnity provided by the path T for data d sent to E_n, that lies in the range [0,1].*

4.4.7 Trusted Path Selection Criteria

The trusted paths for various criteria are represented in the form of a spanning tree with originator as root and all the recipients as leaves. The selection criterion is designed to meet category specific trust requirements without incurring excessive costs. Trust requirements may be specified in terms of maximum path trust, minimum transitive depth, minimum intermediary costs or their combination. When attempting to optimise one variable (such as path trust), limits may be specified for other variables (such as transitive depth or path trust). Figure 4.5 shows two trusted paths created for C_4 category data from *A* using the network shown in Figure 4.3. The one on the left uses minimum transitive depth as the criterion with the minimum path trust (MPT)set to 2, while the one on the right uses the criterion maximum path trust with maximum transitive depth (*MTD*) restricted to 4.

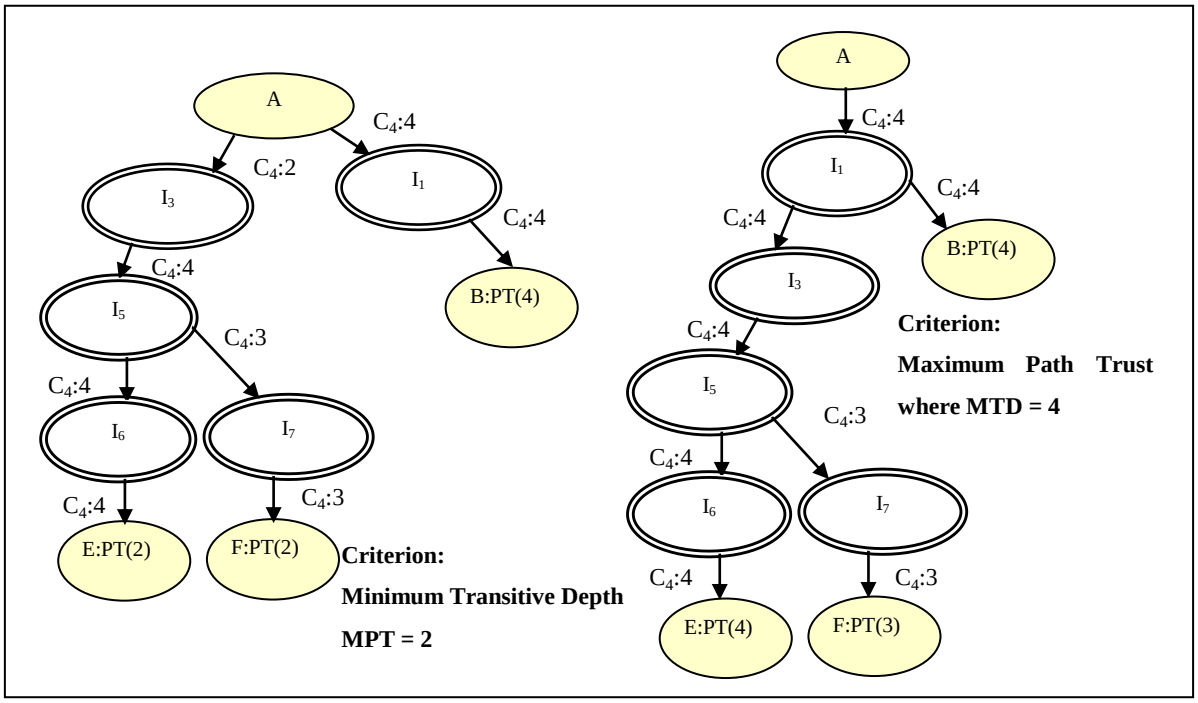


Figure 4.5 Trusted Path between Entity A and all others for the Network in Figure 4.3
Left: Path with Minimum Transitive Depth Right: Path with maximum Path Trust

4.4.8 Objective Criteria for Selecting Trusted Path

Security and trust based mechanisms are needed in various e-commerce domains, including wireless networks and web services. Overcoming constraints specific to a domain requires different trade-offs. PTEI objective criteria for path selection allows weights for maximum path trust, minimum intermediary-cost and minimum-transitive depth to be varied, as shown in Table 4.2. The first criterion aims for short paths with high trust relationships by combining high path trust value with low transitive-depth in equal weights results. Such a criterion is suitable when intermediary costs are negligible compared to the value of data transferred. The second criterion rates transitive depth as the prime factor, followed by cost and path trust, applicable when response time is critical and risks of trust breach are low. The last criterion rates all three factors equally, which is applicable when risks of trust breach and intermediary costs are high and response time is critical. These weights are combined to form the objective search function for trusted path generation.

Quality of Service Factor	**Path Trust (PT) weight**	**Transitive Depth (TD) Weight**	**Intermediary Cost (IC) Weight**
Criteria 1	1	1	
Criteria 2	1	3	2
Criteria 3	1	1	1

Table 4.2 Specifying Objective Criteria for Path Selection

For example, in the network shown in Figure 4.3, assume that *A* must send a category 4 (*C4*) message to *E*. If *PT* or *PT/TD* is the criterion then intermediary path $I_1 \rightarrow I_3 \rightarrow I_5 \rightarrow I_6$ will be chosen, where *PT* is the path trust and *TD* is the transitive depth measuring the number of intermediaries. If the criteria is *1/TD* or PT/TD^2 then the path $I_3 \rightarrow I_5 \rightarrow I_6$ will be chosen instead.

DEFINITION 4.10 OBJECTIVE CRITERIA FOR PATH SELECTION

Objective Critera $fCrit = \dfrac{PT^i}{TD^j.IC^k}$ *speifies criteria for selecting the nodes along the trusted path using weights i,j,k : N for path trust PT, transitive depth TD and intermediary costs IC.*

Each node along a trusted path is linked to the previous node which provides the highest objective criteria for that node. Note *fCrit* uses inverse values of *TD* and *IC* to minimise the number and overall costs of intermediaries. The actual objective criteria specified must also consider the underlying trust coupling (Terminology in Section 4.3), the type of intermediaries, security concerns and cost constraints.

4.4.9 Setting Maximum Transitive-Depth

PTEI sets the maximum transitive depth (*MTD*) for a domain reflecting trust coupling (refer to Definition 4.2), and the number of entities in the domain. Intuitively, the higher the trust coupling in a domain the lower the number of intermediaries required. For example, if there are 100 entities, with each entity having a relationship with 10 others on average, each entity may only require 2 intermediaries on average, as $10^2 = 100$. It is evident that the number of entities that can be reached transitively grows exponentially, suggesting *MTD* should be of $O(log_{TC} N)$.

4.4.10 Trust Transfer and Trust Evolution

The institutional framework alone cannot dispel the perception of risk in traditional commerce. For example, Alice, with low trust disposition for dental work, may need many recommendations before she visits the dentist who has recently moved to her neighbourhood, even though he is highly qualified (endorsed by certified authorities). However, once the initial trust is established, the extent of trust relationship is likely to grow with positive experiences (treatments). Thus, individual experiences and recommendations, too, have played a major role in shaping trust in traditional commerce. If such trust relationships are to be dynamic, the underlying mechanisms must allow trust networks to be self-regulated. PTEI uses trust evolution and trust transfer mechanisms described below.

4.4.10.1 Network Initialisation

The initialisation of a trust network requires each entity to specify the extent to which it trusts another for a specific knowledge category. When two intermediaries or an intermediary and a trading entity express the desire for complementary roles between them for a specific category (one as a sender and the other as receiver) with trust values ranging from *TRMIN* to *TRMAX*, a directed edge from sender to receiver is added to the network. The trust relationship value for this edge is set to the lower of the two trust values specified by sender and receiver.

4.4.10.2 Trust Transfer Policies

Trust transfer policies in PTEI allow new trust relationships to be established based on the trust disposition of individual entities and intermediaries. Reputation is the sum of products of existing endorsement trust relationships with common neighbours. *A*'s reputation in *B* for being a trustworthy sender (refer to Figure 4.6) is computed the same way as *B*'s reputation in *A* for being a trustworthy recipient.

DEFINITION 4.11 REPUTATION

Let $E_1, E_2, \ldots E_n$ *are common intermediaries shared by entities* E_A *and* E_B.

$TR(X,Y)$ *be the endorsement trust relationship between X and Y.*

then $Reputation(E_A, E_B) = \sum_{i=1}^{n} TR(E_A E_i) \times TR(E_i E_B)$

$Reputation(E_A, E_B)$ *represents* E_A*'s reputation in* E_B *for being a trustworthy sender and* E_B*'s reputation in* E_A *for being a trustworthy recipient*

Note that reputation in this model is a directed relationship, that is, reputation for *A* to a sender and *B* to be a receiver is different from *B* to be a sender and *A* to be a receiver. By using the product of trust relationships for computing reputation, trust transfer policies weigh the extent of trust relationships higher than the number of trust relationships.

Example:

A directed edge is added to the network when the computed reputation between two entities exceeds the trust transfer thresholds for both entities. For the network shown in Figure 4.6, a new direct trust relationship for category C_i can be formed between sender *A* and receiver *B* if the required trust transfer thresholds (which reflect their trust disposition) for both *A* and *B* are less than *9 (3×1 + 1 × 2 + 2 × 2)*. This formula is similar to the global reputation scores used in P2P communities [54].

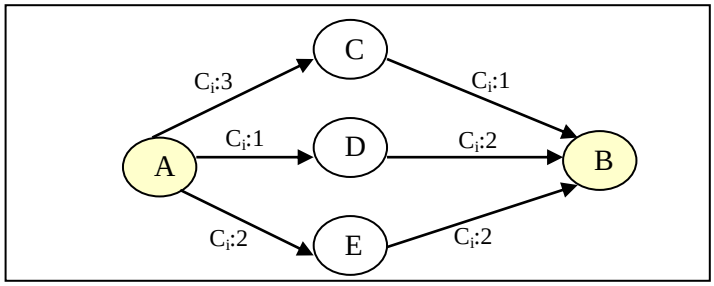


Figure 4.6 Trust Network where Reputation between Sender A and Receiver B is 9.

Though entities with high trust disposition (hence low trust transfer threshold) may form a greater number of links, they also run the risk of severe damage to existing trust relationships by being associated with unreliable intermediaries. Unreliable intermediaries are those with high likelihood of trust breaches and invalid endorsements. Therefore the optimal trust transfer threshold for any domain must be determined experimentally reflecting the trust degradation rate *TDR*, the trust severance period *TSP* and the average reliability of entities in the domain.

4.4.10.3 Trust Evolution Policies

Trust Evolution policies in PTEI determine how success or failure of transactions impacts trust relationships. Successful transactions cause growth in trust relationships based on the value for domain

TER. Some recent models have proposed context factors such as the transaction value should be included when modelling trust. Trust growth in PTEI can also be made a function of transaction value to prevent malicious entities building up trust using low valued transactions [160]. Trust relationships decline sharply when trust assumptions are breached based on domain *TDP*. The extent of decline reflects the distance (in hops) from the point of breach. Figure 4.7 shows the impact of trust breach by node E on trust relationships along the path, assuming the trust degradation rate (*TDR*) is 0.5. All trust relationships with entity breaching trust are severed for the period TSP (trust severance period) specified by domain policies.

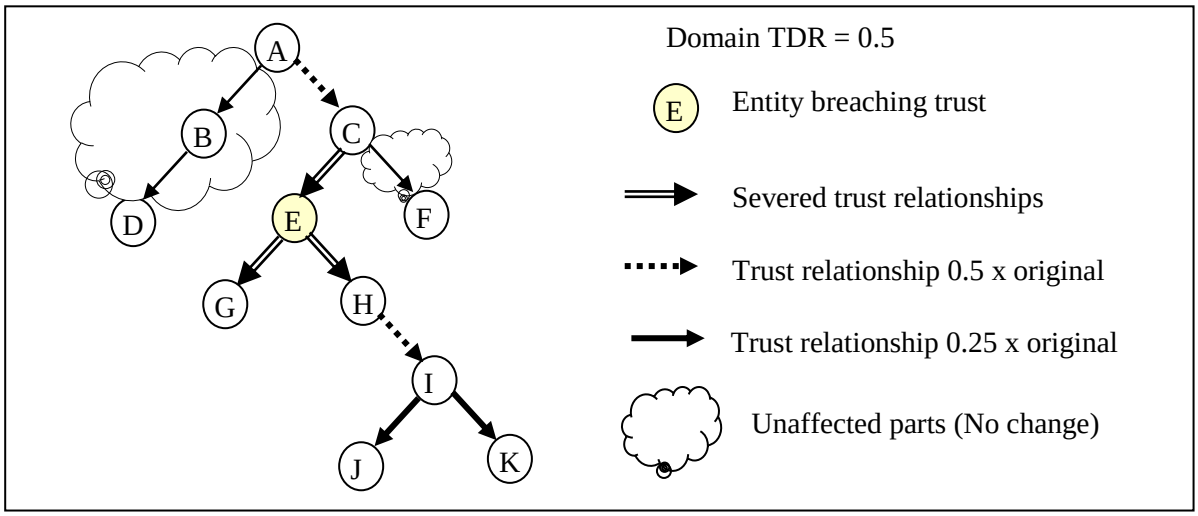


Figure 4.7 Trust Degradation along the Trusted Path from E's Trust Breach

4.5 Internal Representation and Algorithms

If e-commerce trusted paths are to be computed at runtime, trusted path generation and retrieval algorithms must be scalable. The search criteria used must be made flexible if trade-offs between path trust and costs are to be achieved for different environments and values of messages. Section 4.5.1 presents techniques devised to meet overall design goals. Section 4.5.2 presents the internal representation of various components, while Section 4.5.3 outlines the main algorithms devised.

4.5.1 Design Techniques to Address PTEI Implementation Goals

PTEI implementation goals relate to flexibility in specifying trust requirements, low response-time, acceptable performance for large networks and modelling trust.

- Faster response-time for transactions relies on efficient trusted path creation/retrieval mechanisms. Domain trusted paths for various combinations of attributes are pre-created and updated regularly in PTEI as trust relationships are constantly evolving. The underlying PTEI algorithms are made efficient by using a form of spanning tree, which allows trusted paths to be maintained for all recipients.
- Flexible objective criteria (*fCrit*) for path selection to allow the right trade-offs for different environments.

- The growth in the size of a network can adversely affect the update and search time for trusted paths. PTEI reduces search time by subdividing large networks into hierarchically organised domains. Each such domain contains a number of pre-created sub-trees to allow efficient trusted path retrieval. Sub-trees are created for various combinations of message origin, category and maximum transitive depth.
- PTEI self-regulates the trust network through trust evolution and transfer policies. These policies aim to cause growth in the number and extent of trust relationships for reliable intermediaries, and reduction in trust relationships with unreliable intermediaries.

4.5.2 Internal Representation using Domain Trees and Sub-trees

Trading entities and intermediaries are divided into hierarchical domains allowing easy access to trusted paths. The domain tree is divided into sub-trees for each source node in that domain, which in turn are subdivided repeatedly for each data category, objective criterion and maximum transitive-depth (*MTD*), as shown if Figure 4.8. The final (terminal) sub-tree is a spanning tree that optimises the trusted path from the specified source to all other domain destinations, using the criterion specified.

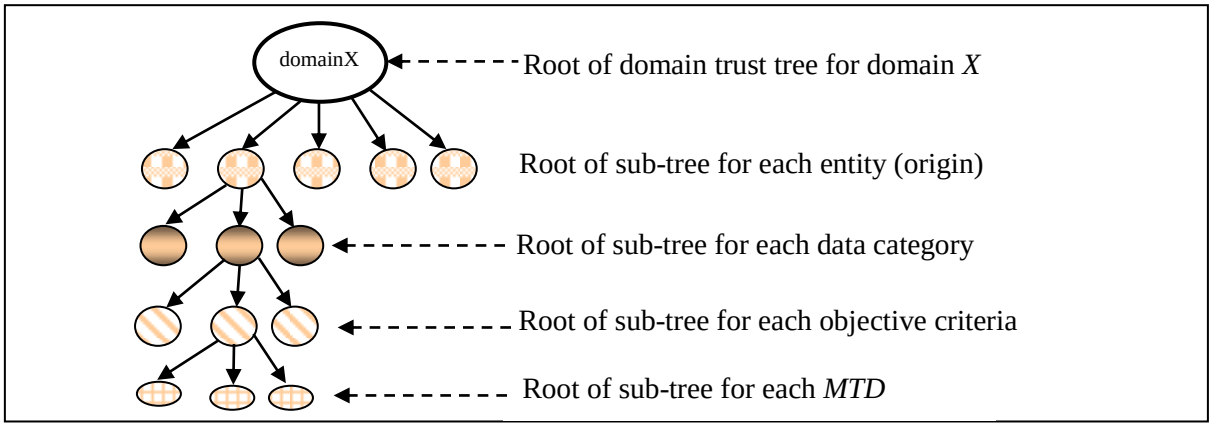


Figure 4.8 Organisation of Domain Trust Tree

4.5.2.1 Worst Case Memory Requirement Analysis

The worst case domain memory requirement (*DMReq*) for each domain can be computed on the basis of domain-size in bytes (*DS*), maximum number of data categories (DC_{max}), number of different objective criteria (OC_{max}) and size of tree node *NS* as $MReq = DS \times DS \times DC_{max} \times OC_{max} \times NS \times MaxMTD$. Note domain-size (*DS*) is used twice as it determines the number of root nodes and the maximum number of nodes in the tree. Also each domain may allow trusted paths to be generated varying *MTD* from 1 to *MaxMTD*. The actual memory requirements are much smaller as only entities with the required endorsement capabilities can be part of a sub-tree making the number of nodes much less than the domain size. Thus for a domain size of 200 entities, 5 different objective criteria, 10 data categories, 5 different values for *MTD* and node size of 200 bytes the worst case memory requirement is only 2GB allowing the entire data structure to be stored in memory. Moreover, the availability of inexpensive large scale memory makes the cost of memory less critical than response time, in the design of trust servers.

4.5.2.2 Minimising Search and Retrieval Times

This section presents the data structure devised to minimise the trusted path search and retrieval times, using an example. Figure 4.9 shows the sub-tree that maximizes path trust for a category *{C4}* message from *A*, for the trust network shown in Figure 4.3. The trusted path from *A* to entities *B*, *E* and *F* uses intermediaries I_1, I_3, I_5, I_6 and I_7. The root of this sub-tree contains a label for all reachable entities together with corresponding path trust and transitive depth. All intermediate nodes (intermediaries) are also labelled with entities that can be reached, to reduce retrieval time. For example, the set *{E,F}* is set as *dests* for intermediary I_3 as it lies along the path from *A* to *E* and *F*.

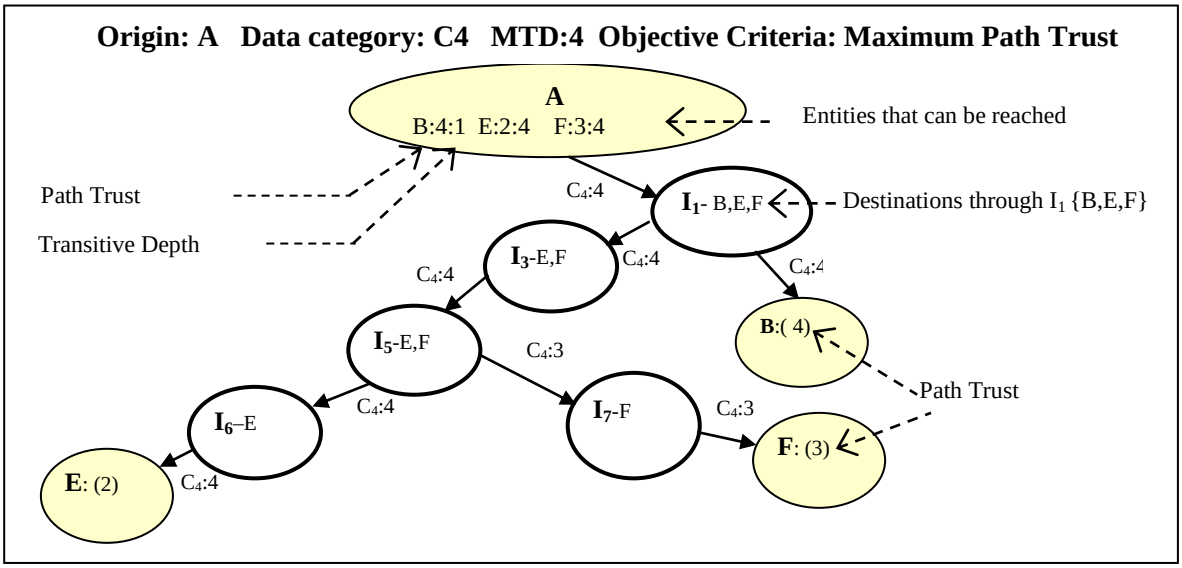


Figure 4.9 Root of Sub-Tree Labelled with Trustworthiness and Transitive Depth for Each Entity

4.5.3 PTEI Algorithms, Policies and Schemes

This section presents the main algorithms used in PTEI. Section 4.5.3.1 describes the features of these algorithms. Section 4.5.3.2 presents the algorithms used for creating domain sub-trees. *DSTGen* used for generating the domain sub-tree is a variation of Prim's spanning tree algorithm that can be executed in linear time [140]. Section 4.5.3.3 presents the algorithm for retrieving the trusted path for a message from a given source to a specific set of entities efficiently. Section 4.5.3.4 presents the schemes used for retrieving trusted paths in large networks efficiently using a hierarchical structure of domains. All these experiments were carried out using a 2.5 GHz laptop with 4GB RAM running Windows XP operating systems.

4.5.3.1 Features of PTEI Algorithm

PTEI features are designed to make the algorithms scalable and provide quick response times. These features are briefly described here.

- No data is passed through an intermediary more than once, allowing an upper limit for the number of possible edges in a domain trust tree in terms of number of entities, data categories and objective criteria for path selection.

- Limiting the maximum number of intermediaries makes the algorithm scalable. Any search is abandoned if a trusted path cannot be found within *MTD* number of intermediaries.
- Different sub-trees are created and stored for different trust/performance criteria. The root of each such sub-tree contains all the entities that can be reached, allowing efficient search for trusted paths based on trust/performance criteria and required destinations.
- The algorithm retrieves trusted paths from existing domain trust trees in linear time, if one exists.

4.5.3.2 Algorithm for Generating and Storing Domain Sub-Tree

DSTGen computes the optimal spanning tree from a source to all other entities in the domain using the objective criteria specified for the data category. Only intermediaries with the required endorsement capability are added to the spanning tree. The number of intermediaries along the spanning tree is limited by the maximum transitive depth (*MTD*) specified. *DSTStore* stores the domain sub-trees generated by DSTGen. These algorithms are implemented in Java using around 600 lines of code.

DSTGen takes as arguments *dc* the data category, *O* the data origin, *trels* the 2-dimensional array storing all the trust relationships in the network, *fCrit* the objective function for trusted path selection and the maximum transitive depth allowed *MTD*. The class *node* used for storing entities lying along the trusted path, has attributes *fCrit*, current depth and a reference to its parent node. The algorithm creates the spanning tree *pTree* rooted in *O* using the trust relationships stored in *trels*. The set *unused* stores all nodes representing entities currently not lying along the trusted path. If *trels* contains a link between the current *pTree* node and an *unused* node with endorsement capability *dc*, the *unused* node is added to *pTree*. If *trels* has a link that improves *fCrit* for a given node, the sub-tree rooted in that node is rotated, provided the number of intermediaries does not exceed *MTD*. Note the algorithm allows at most one node per entity.

DSTGen(dc, O, trels, fCrit, MTD)

1. set root *pTree* ← *O*
2. set *unused* ← entities \ {*O}*
3. if *trels* has an element that links a *pTree* node *e1* with less than *MTD* intermediaries to an *unused* node *e2* with endorsement capability *d,* add *e2* as a child node of *e1* and remove *e2* from *unused*
4. otherwise, if *trels* has an element that links a *pTree* node *e1* to another *pTree* node *e2* and the path through *e1* entity improves *fCrit* for *e2* entity but without causing the number of *pTree* intermediaries to exceed *MTD*, move the sub-tree rooted in *e2* as a child element of the node for *e1*. Update *fCrit* and depth for all elements in that sub-tree
5. repeat steps 3 and 4 until no links can be found from an entity in *pTree* to any other *unused* entity or between entities in *pTree* that provides improved value for a specified criterion
6. return *pTree* after updating all nodes for destinations reached through them

DSTStore stores the domain sub-trees for various combinations of data origin, category, *MTD* and *fCrit* in a hash using the *DSTGen* algorithm presented above. Data category can be any subset of the set of categories *C* and can originate from any of the entities in *EN*, the set of all domain entities. It is also assumed that *MTD* can be allowed to vary up to *MaxMTD* with f*Crit* selected from the set of all objective functions *fCritSet*. The current set of trust relationships in the network is stored in the array *trels*.

DSTStore(trels,C,fCritSet,E,MaxMTD)

```
for each dc ∈ C
  for each fCrit ∈ fCritSet
    for each O ∈ E
      for MTD = 1 to MaxMTD
        subtree(dc,O,fCrit,MTD) ← DSTGen(dc, O, trels, fCrit, MTD)
```

4.5.3.3 Algorithm for Trusted Path Retrieval

The algorithm *TrustPathRetriever* retrieves a domain sub-tree using data origin, data category, objective function for trusted path selection and the lowest transitive trust depth possible. The function *GetTrustPathTree* creates the trusted path to all the recipient (destination) nodes in the form of a spanning tree. These algorithms were implemented in Java using around 350 lines of code.

The algorithm *TrustPathRetriever* takes as arguments the data category *dc*, the origin *O*, the objective function *fCrit* destinations *dests* and the maximum transitive depth (*MAXMTD*) allowed. It returns the sub-tree with the lowest *MTD* that meets the requirements by varying *MTD* from 1 to *MAXMTD*. If no such subtree can be found it returns *null*. Otherwise it returns the trusted path generated using the *SetTrustPath* function.

TrustPathRetriever(dc,O,fCrit,dests,MAXMTD)

```
for MTD from 1 to MAXMTD
   st ← subtree( dc, O, fCrit, MTD)          // domain trust tree with required parameters
   if ( dests ⊆ st.dests )                   // trusted path contains all required destinations
      tp ← getTrustPathTree(st.root, tp.root, dests);    // forms the nodes
      return tp;                             // return trusted path
   else
      return null;                           // no domain sub-tree with required destinations)
```

GetTrustPathTree is a recursive function that takes as argument domain sub-tree or one of its branches named *dsTree* and the required data destinations *dests*. Each node of *dsTree* contains a label

named dests specifying all destinations that can be reached through that node. *GetTrustPathTree* returns the spanning tree formed combining all branches leading to any of the required destinations.

GetTrustPathTree(*dsTree, dests*)

```
tp = ∅                                          // initialize trusted path sub-tree
set tp.root ← dests.root
for each branch ∈ dests
  if (branch.root.next ∩ dests ≠ ∅)             // if this branch leads to one of the destinations
    tp.add(GetTrustPathTree(branch,dests))      // add that sub-tree
return tp;
```

4.5.3.4 Efficient Trusted Path Retrieval Using Hierarchical Domains

The first part of this section presents the details of hierarchical domains used to retrieve trusted paths for large networks. The techniques for two-level hierarchies are presented first as shown in Figure 4.10, before extending them to multiple hierarchies. The key design features and the rationale are outlined below.

- When the number of entities in a trust network grows hierarchical domains can be formed to reduce the memory requirements and the path generation time. Memory requirements for maintaining trust relationships can grow at polynomial rate as n entities can have n^2 number of trust relationships. Path generation time can be reduced significantly as the search path requires less traversing.
- With hierarchical domains maximum transitive depth can be used to restrict the trading domain. When it is low only locally matching entities can be found while higher levels can be used to expand the trading domain.
- Hierarchical domains many require subdividing a global problem into multiple domain based problems. However, the sum of locally optimal solutions may not produce the globally optimal solution. The domain size therefore must be set reflecting the need for globally optimal solutions and the memory capacity of servers. With decreasing cost of memory and fast processers the domain sizes can be increased far beyond the domain size used in this thesis.
- Entity identifiers are made unique by combining domain and entity names. In Figure 4.9 the high-level domain is labelled *HI*. Each low-level domain consists of at least one shared entity (part of both low and high level domains) with the capacity to endorse all entities in the low level domain, thus forming a hierarchical endorsement structure. Such shared entities have multiple identifiers such as *HI-M/1-W*. The hierarchical structure can be extended to any number of levels.
- The task of sending a message to n entities from different domains can be subdivided into multiple intra-domain sub-tasks. For example, in Figure 4.10 entity *3-D* sending a message to destinations {*3-C, 1-Y, 2-E, 2-F*} can be subdivided into the following intra-domain sub-tasks:

{*3-D*} sending a message to {*3-C, HI-N/3-A*}, {*HI-N/3-A* } sending a message to {*HI-M/1-W*} and {*HI-O/2-H*}, {*HI-M/1-W*} sending a message to {*1-Y*} and {*HI-O/2-H*} sending a message to {*2-E, 2-F*}.

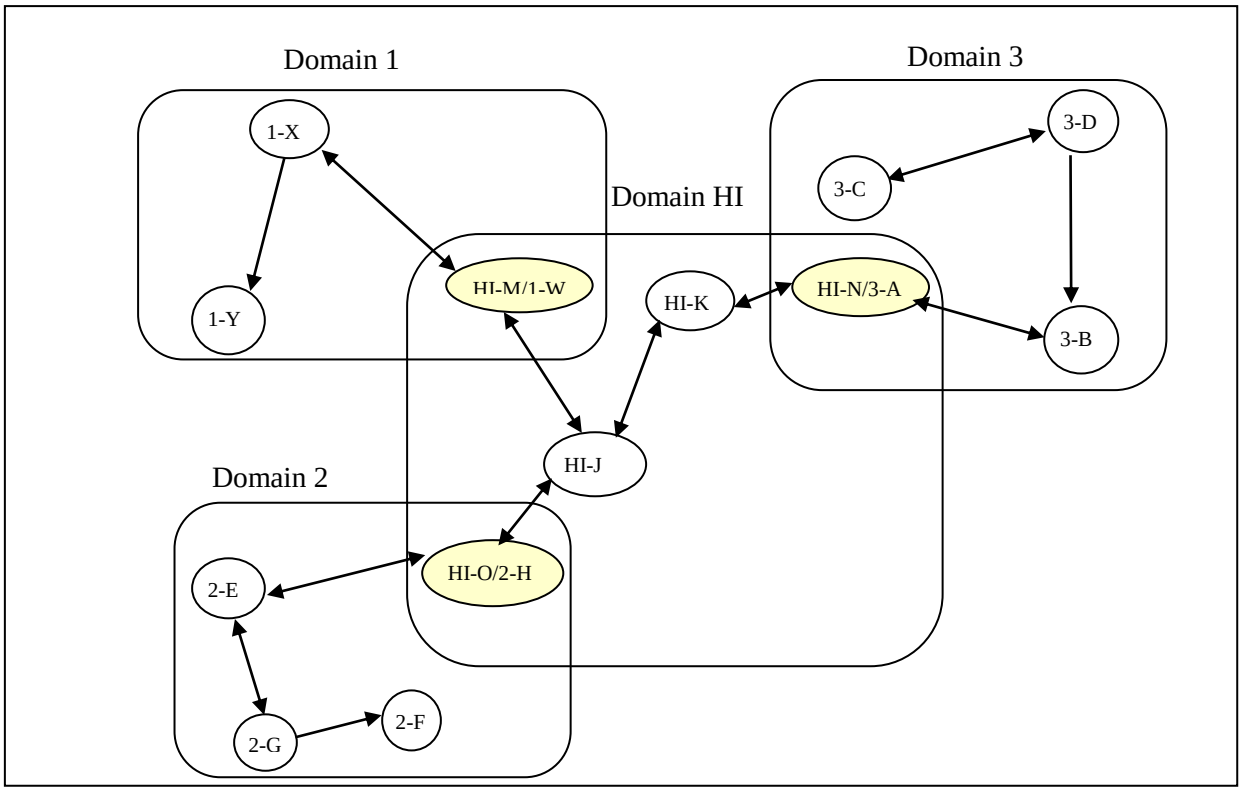


Figure 4.10 Hierarchical Clustering of Domains

Complexity Analysis of Path Retrieval

This section briefly analyses the complexity of the multi-domain search algorithm. Assume the number of entities in the network is *n*, and the predefined domain size is *s*, and the average transitive depth for the domain is *td*. Assuming the whole trust network of entities is organised into a balanced tree of domains, the number of hierarchical levels (*l*) is $log_s n$. For any message sent to *d* destinations: the maximum number of steps occurs when every recipient is found in a different domain at the lowest level (which involves searching all the way to the top and then back). Hence, the maximum number of steps is *2dl*, which grow linearly with *d*. Hence using such a clustering mechanism the algorithm can be made scalable. Furthermore, the clustering mechanism allows the global search problem to be subdivided into tasks that can be handled concurrently by different trust servers.

4.6 Experimental Results & Comparison with Other Work

This section is designed to verify a number of hypotheses relevant to the design of PTEI framework. By considering the factors that cause trust growth or decline, the centralized trust network in PTEI allows trusted paths to be generated much faster than distributed networks where each node maintains its own trust relationships. This section attempts to verify the effectiveness of PTEI trust policies in modelling factors that are known to influence trust, such as experiential trust and trust disposition [39,41]. All trusted paths in PTEI are depth limited, though the underlying selection criteria can be made to vary combining the extent of trust relationships, transitive depth and endorsement costs. The impact of these criteria in networks with different trust couplings are compared in this section. This section also verifies the effectiveness of the underlying architecture and path finding techniques for large networks. The table below gives the rationale and the hypotheses to be tested. All of these hypotheses are tested using a simulation approach using randomly generated trust network.

Rationale	Hypotheses to be Tested
Trust disposition is known to be a major factor that causes growth in trust relationships. In Traditional commerce, traders with high trust disposition and are willing to take greater risks may benefit up to a point. Business confidence may decline if traders take too high a risk.	If trust transfer threshold modelling trust disposition is valid, a similar relationship (in the shape of an inverted parabola) must exist between trust transfer threshold and trust spread (which models trust capital). Note low trust transfer threshold corresponds to high trust disposition, and vice-versa.
In traditional commerce positive customer experiences and the resulting trust capital plays a major role in business growth.	If experiential trust is modelled correctly a linear relationship must exist between node reliability or trustworthiness and trust spread.
A number of factors can affect the elapsed time for path generation. PTEI domain sizes should be set to allow path generation at runtime.	Elapsed time should vary significantly with the level of coupling and grow at a polynomial rate for the highest coupling where every entity has a relationship with another.
When depth limited trusted paths are selected on the sole criteria of highest trust relationship, trusted paths may not exist for many entities.	When transitive depth is limited, significantly more trusted paths can be established when the objective function combines transitive depth with extent of trust relationships (compared to using trust relationships alone).
Trusted paths must be created in real time if e-commerce entities are to interact.	Algorithms can be devised to retrieve trusted paths from domain trust trees that are linear in complexity for varying domain sizes and number of recipients.

Section 4.6.1 verifies the first hypothesis by measuring the impact on trust growth for varying trust transfer thresholds. This section also attempts to verify the second hypothesis by measuring the impact of trust evolution policy on entities with varying degrees of reliability. Sections 4.6.2 is designed to test the third hypothesis by comparing the elapsed time for trusted path generation with varying levels of trust coupling. Objective functions for generating trusted paths are specified in terms of low transitive depth, low intermediary cost and high path trust. Section 4.6.3 tests the fourth hypothesis by varying the weights attached to these factors and comparing their impact on the percentage of valid trusted paths formed. Section 4.6.4 tests the fifth hypothesis by measuring the path retrieval times. Section 4.6.5 compares the rate of trust attenuation in the existing trust models with the endorsement trust model proposed. All these experiments were carried out using a 2.5 GHz laptop with 4GB RAM running Windows XP operating systems. All the algorithms and simulations were implemented using Java with the lines of code varying from 200 to 400 lines of code.

4.6.1 Trust Evolution and Trust Transfer Parameters

The experiments in this section are designed to measure the impact of reliability on overall trust relationships through trust evolution policies. It also measures how the trust transfer threshold (Definition 4.6) impacts overall trust relationships through trust transfer policies. The experiments were carried by simulating a trust network with 200 entities. The overall trust relationship of an entity is measured using Trust Spread which is the sum of trust placed on an entity, as defined below.

DEFINITION 4.12 TRUST SPREAD

$$Trust\ Spread(E_i) = \sum_{j=1}^{n} TR(E_j E_i) + \sum_{j=1}^{n} TR(E_i E_j)$$

where $TR(E_j\ E_i)$ is the trust relationship between entities Ej and Ei, and n is the number of network entities.

Figure 4.11 shows the effect of PTEI trust transfer and evolution policies on trust disposition by measuring trust spread against trust transfer threshold, where all entity reliabilities (Terminology in Section 4.3) are set at 40,000. The penalties and rewards for entity conduct are also kept the same for all entities. It is evident that PTEI policies are effective is dissuading entities from setting the trust disposition either too low or too high.

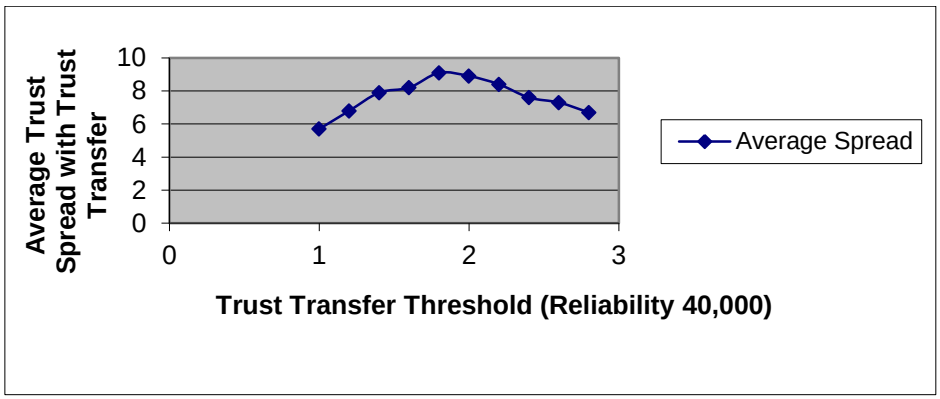


Figure 4.11 Trust Spread with Varying Trust Transfer Threshold

Results and Analysis

In Figure 4.11, the trust spread increases when trust threshold increases from 1 to 1.8, because the likelihood of trust breaches through unreliable associations is reduced. However, beyond the optimal point of 1.8, trust spread decreases as the chances of forming new trust relationships (both good and bad are reduced). The experimental results also show the optimal point for trust threshold varies significantly when trust parameters reflecting risks and rewards are varied. Hence, the optimal parameters should be set considering the underlying domains and categories.

Figure 4.12 shows the impact of trust evolution policies (Section 4.4.10.3), which specify rewards and penalties for positive and negative conduct respectively. Initially all entities in the network are connected to at least one other entity, with trust relationship value of 1. The experiment regenerated the domain trust trees 100 times using the updated values of trust relationships to measure the impact of these policies over time. Messages are sent up to 1000 times or until trust is breached (simulated based on the reliability) along the trusted paths in domain trusted-trees, updating the trust value each time. Penalty for trust breach (stated in terms of period of distrust following a trust breach) is kept constant across the domain.

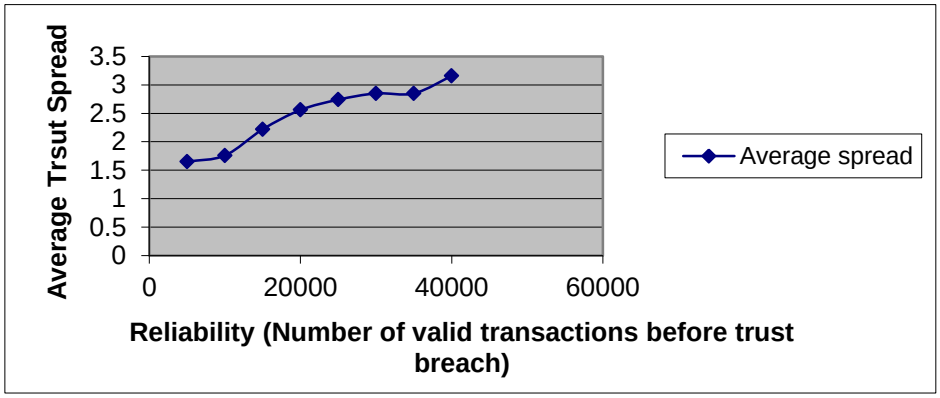


Figure 4.12 Trust Spread through Trust Evolution Policies for Varying Reliability

Results and Analysis

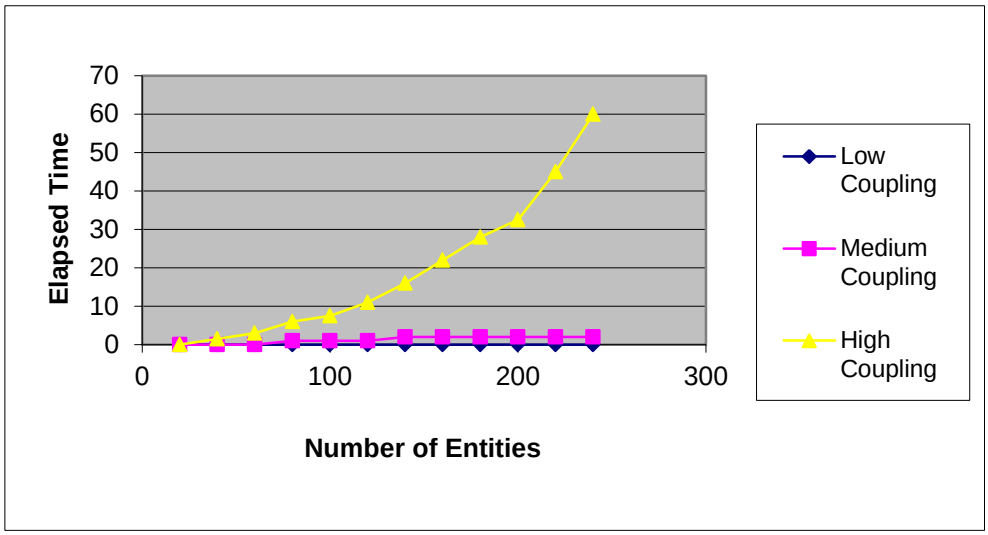
The results of measuring trust spread with varying trust transfer thresholds as depicted in Figure 4.11suggests the trust transfer policy is effective in modelling trust disposition. Figure 4.12 shows the relationship between reliability and trust-spread in nearly linear with a correlation coefficient of 0.92.

The increase in trust spread is caused by both trust evolution and trust transfer policies. The more reliable entities are less likely cause a trust breach or endorsement failure leading to trust devaluation. The effect of trust degradation is not evident in the graph as these results are obtained averaging the results for different trees and data paths. These results show policies for maintaining PTEI centralized trust network are effective in modelling "business growth in traditional commerce through trust disposition and positive customer experiences".

4.6.2 Impact of Trust Coupling on Performance

Trust coupling (Terminology in Section 4.3) reflects the average number of trust relationships an entity has with others. Assuming each entity has a (directed) trust relationship with at least one entity, the number of one-way trust relationships in a domain of 200 entities can vary between 200 and 40,000. In this section the effect of trust coupling on performance is measured using high, average and low coupling. High coupling assumes that every entity trusts all others in the domain, meaning the average number of trust relationships is the same as the number of entities. Highly coupled networks are unlikely to be encountered in real life except in small domains. In lowly coupled networks, each entity trusts only one other entity for sending or receiving a message. Lowly coupled networks are also uncommon, except in mobile ad-hoc networks where each entity may be allowed to send or receive a message from only one other entity [141]. Based on empirical studies in social networks, the average number of intermediaries for medium coupling is set at 2 [142]. In order to reach all entities in a network of *n* nodes using only 2 intermediaries, the average trust coupling therefore must be $n^{½}$ [142]. For example, if medium coupling is used an entity which has 10 trust relationships in a community of 100, will end up with 20 trust relationships when the community size increases to 400.

Networks with high, medium and low coupling were simulated using normal distribution of trust relationships varying from 1 to *TRMAX*. The maximum transitive depth was restricted, as cost and bandwidth increases with each additional intermediary, and opens more venues for attacks; it was set at 5 based on empirical studies for social networks of a similar size [143]. It was shown that 90% of recipients in networks of mixed coupling can be reached with 5 or less intermediaries [143]. Figure 4.13 shows the algorithm performance where the number of entities in the network varies between 20 and 240. Elapsed time shows acceptable performance (less than 70 milliseconds), though it grows linearly for low and medium coupling and at polynomial rate for the highest coupling. These differences directly reflect the number of trust links. Trusted paths for larger networks are generated within acceptable bounds by subdividing them into hierarchical domains (Section 4.5.3.4) and by computing trusted paths offline (Section 4.5.3.3).



	20	40	60	80	100	120	140	160	180	200	220	240
Low	0	0	0	0	0	0	0	0	0	0	0	0
Medium	0	0	0	0	1	1	2	2	2	2	2	2
High	0	1.5	3	6	7.5	11	16	22	28	32.5	45	60

Figure 4.13 Elapsed Time for Low, Medium and High Coupling (LNHMTD5Res)

Results and Analysis

Table 4.3 shows percentage of recipients for which valid trusted paths can be found (for a fixed maximum transitive depth) varies significantly with coupling levels. Though low and medium coupled networks scale well with the number of entities, the performance of highly coupled network degrades significantly. This can be attributed to the underlying data, which models the worst-case scenario for high coupling where every entity in the domain has a trust relationship with each other. Even with this worst case scenario, trusted paths can be found within acceptable time bounds as the network is subdivided into hierarchical domains where the size of domain is restricted.

Though the low and medium coupled networks have better performance, Table 4.3 shows that the percentage of valid trusted paths is significantly lower than for highly coupled networks. The problem is exacerbated for large networks. This is because limited transitive depth impacts larger networks more than smaller networks.

Number of Entities	High Coupling	Medium Coupling	Low Coupling
20	100%	90%	56.5%
40	100%	90%	39%
60	100%	88.5%	30%
80	100%	86%	21%
100	100%	86%	18.5%
120	100%	83.5%	17%
140	100%	81%	13%
160	100%	80.5%	13%
180	100%	81%	11%
200	100%	79.5%	11%
220	100%	79.5	8.5%
240	100%	79.5	8.5%

Table 4.3 Recipients with Valid Trusted Paths

4.6.3 Path Selection

Criteria for path selection are made to reflect the number of intermediaries and the extent of trust relationships between trading entities. This section measures the impact of varying the relative weights attached to path trust (*PT*) transitive depth (*TD*) for a network of medium coupling. Figure 4.14 shows how *PT*, *PT/TD* and PT/TD^2 affect the percentage of trust graphs that can be formed. The maximum transitive depth is set to 5 based on work done with social networks [143].

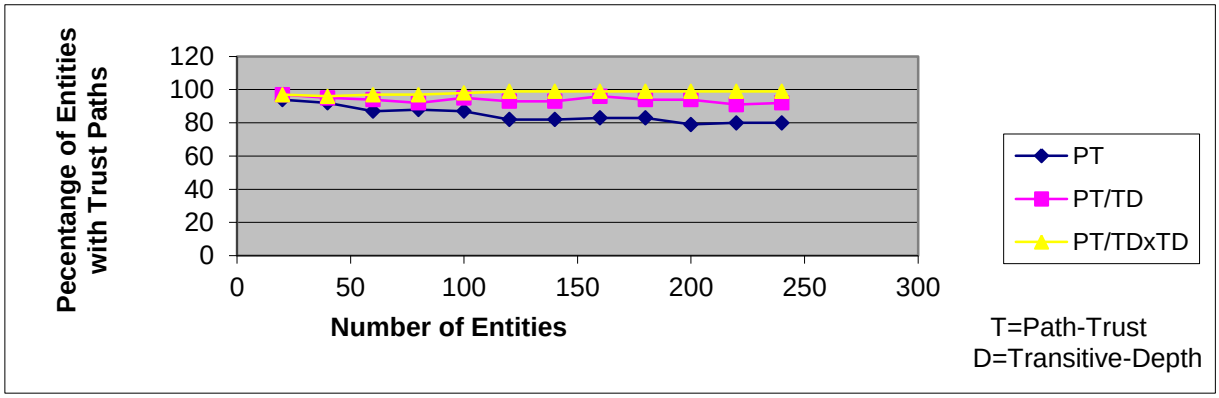


Figure 4.14 Percentage Entities Finding Trusted Paths for Varying Criteria

Results and Analysis

Using path trust (*PT*) alone as the selection criterion seems adequate for finding trusted paths when the number of entities is small (less than 10). However, when the number of entities exceeds 150 the percentage of entities with trusted paths drops significantly. This is because the transitive depth specified does not allow the highest path trust to be attained by all data recipients. With objective criteria PT/TD^2 up to 20% increase is obtained for number of entities exceeding 200. This can be explained as this objective-criteria selects entities based on closer proximity as well as trust as opposed to using trust alone. Thus, a side effect of minimising the number of intermediaries through objective functions such as PT/TD^2 is that more entities can be reached within the transitive depth limit. On the negative side, the aggregate path trust for recipients will be lower than when using trust as the sole criteria. Hence, the weight attached to transitive depth (*TD*) objective criteria must reflect the limit on transitive depth, the cost of intermediaries and need for high path trust. PTEI also allows the cost of intermediaries to be included explicitly in the selection criteria. Though transitive depth indirectly models endorsement costs, it does not model any variation in endorsement cost of intermediaries. Furthermore, congestion in busy nodes can be averted by assigning them high costs.

4.6.4 Trusted Path Retrieval Time

PTEI facilitates trusted path selection at runtime by generating domain trust trees offline and allowing retrieval at runtime. This section measures the time to retrieve trusted paths from pre-existing domain trust trees for networks of various sizes. The average retrieval times are obtained by repeating the simulation 1000 times for 10 randomly generated highly-coupled networks using a varying number of recipients (between 5 and 10). The results presented in Figure 4.15 shows the algorithm scales linearly, with a coefficient of correlation 0.933.

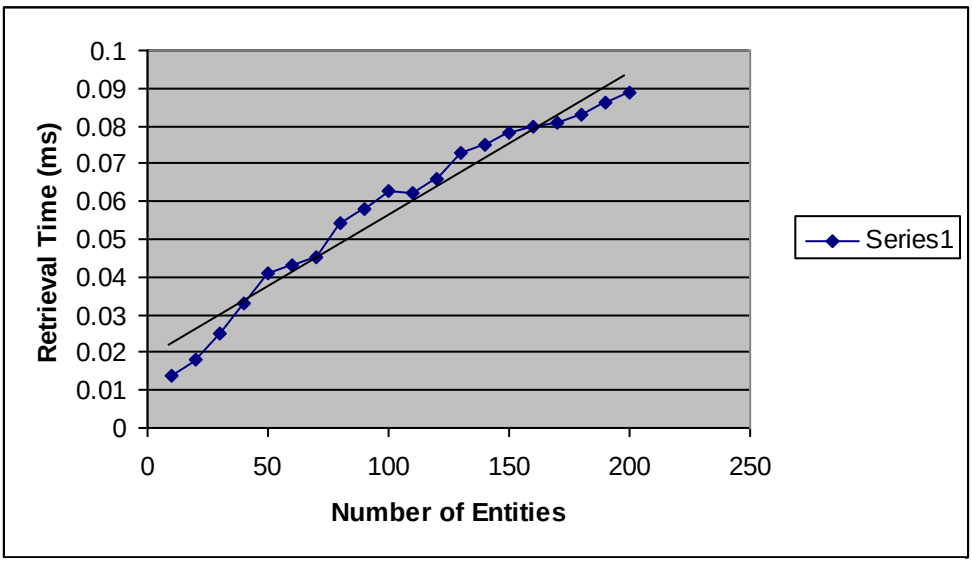


Figure 4.15 Retrieval Time for Varying Number of Parameters

E-commerce applications often require messages to be sent to multiple recipients. Such messages can significantly reduce intermediary costs and bandwidth by using common intermediaries along the path. The next experiment measures the trusted path retrieval time for up to 20 recipients. The results in figure 4.16 show that the algorithm scales linearly with the number of recipients with correlation coefficient 0.95.

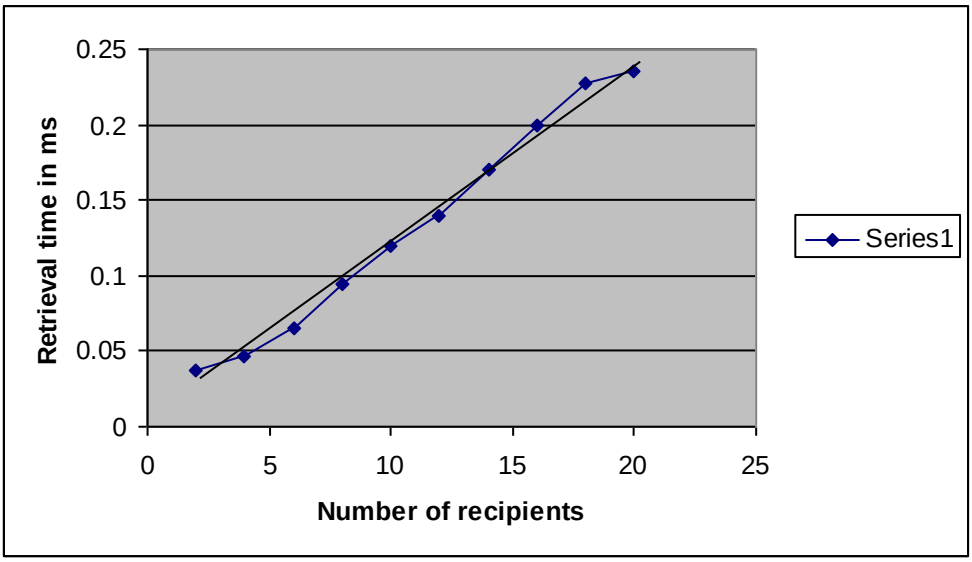


Figure 4.16 Retrieval Time for Varying Number of Recipients

Results and Analysis

The linear growth in retrieval time for highly-coupled networks is obtained in Figure 4.16, because the trust trees are formed offline. Although the linear growth suggests the domain size can be indefinitely increased, the main constraint is the memory required for maintaining separate large trees for every combination of message originator and selection-criteria. Analysis of memory requirements in Section 4.5.2.1 shows that the memory requirement degrades at quadratic rate with the number of domain entities.

Retrieval time also scales linearly with the number of recipients, in Figure 4.16, because each node in trust tree formed offline contains all recipients that can be reached through them (to make retrieval efficient). The low response-time for domain trust trees makes it possible to vary the selection criteria dynamically, to find the right balance between trust and underlying costs.

4.6.5 Comparison with Existing Transitive Trust Models

Social networks model transitive trust relationships as a product of existing trust relationships between successive nodes along the path [133, 136]. Others have set transitive trust value to the minimum trust relationship along the trusted path [129]. The proposed model allows endorsement trust (Definition 4.1) to reflect the minimum trust relationship along the path and the number of intermediaries using the formula *:*

$$PET(M,E_n,T) = \frac{PT(M,En,T)}{TR_{MAX}} \cdot \left(1 - \frac{(TD(T)-1)}{MTD}\right)$$

Figure 4.11 compares these transitive trust models with the endorsement model proposed as the number of hops varies from 1 to 6. It is assumed that the trust relationship between each adjacent node along the path is set to 0.7.

Results and Analysis

Figure 4.17 shows no degradation in transitive trust using minimum trust relationship as it does not consider the number of intermediaries. Using the minimum trust relationship does not model the way trust degrades in real-life where each additional intermediary decreases the trustworthiness of information passed. Modelling trust relationship using the product of existing trust relationships reflects both the trust relationships and the number of intermediaries along the trusted path. However, the transitive trust declines sharply reaching less than 0.2 with five intermediaries. The figure also shows that the rate of decline for the proposed model varies with the maximum transitive depth. When MTD is set to 20 trust degrades at a much lower rate than when it is set to 7. Hence, the proposed model allows the flexibility to vary the rate of decline for different categories and domains by changing the maximum transitive depth (MTD). Varying MTD reflects the intuition that in more trustworthy and less critical environments more endorsement intermediaries may be permitted.

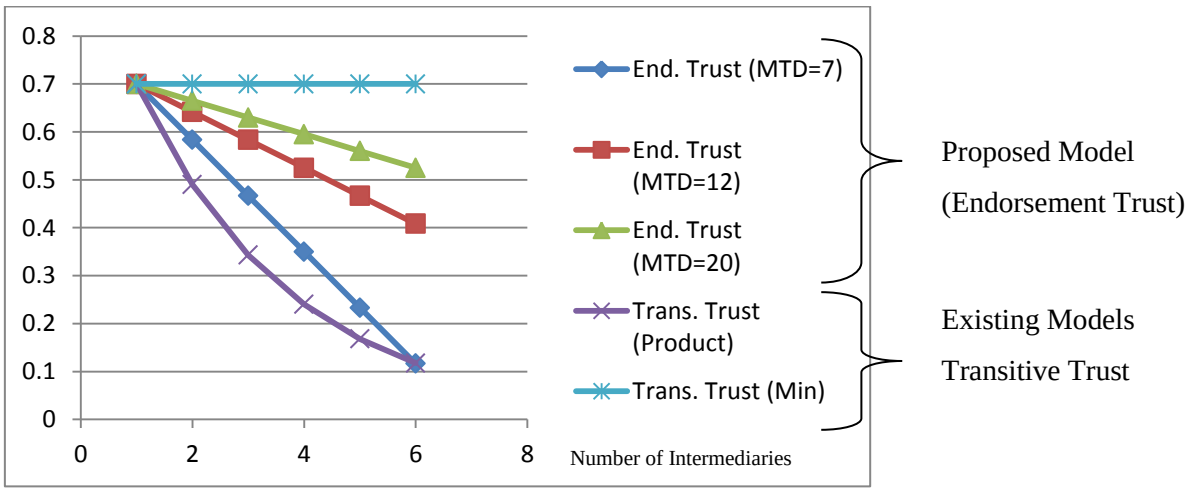


Figure 4.17 Comparison of Trust Attenuation Rate for Endorsement and Transitive Trust

4.6.6 Comparison with Other Trust Models

This section compares PTEI with trust models in other domains. All trust models surveyed allowed transitive depth to be limited when selecting trusted paths as optimal path algorithms are in general considered to be NP-Complete [133]. PTEI allows trust relationships to be classified into categories and organised into hierarchies. Trust in social networks too involves multiple communities organized into hierarchical nodes [134]. However, use of a link structures instead of nodes are necessary to avoid the complexity introduced by overlapping hierarchies [134]. PTEI uses a global trust network to allow faster generation of trusted paths. However, such an approach may introduce traffic congestions in certain nodes. Another distributed approach, which allows trusted paths to be extended iteratively until a path is found [128], allows greater flexibility in selection criteria for individual nodes. Unlike referral networks [125] which only allow trust evolution, PTEI policies allows trust evolution to be combined with trust propagation, thus allowing establishment of new trust relationships and reduction in transitive depths required. Trust models used in ad-hoc networks allows multiple objectives (trust values and the number of hops) to reduce the hazards from malicious nodes [164]. PTEI too allows multiple criteria such as transitive depth, endorsement costs and path trusts in different weights. The Table 4.4 compares the features of common trust models surveyed.

Trust Model	**Category Specific Trust**	**Hierarchical Trust Relationships**	**Global/Local Trust Model**	**Formation of New Trust relationships**	**Multiple Criteria Allowed**
PTEI	yes	Yes	Global	yes	Yes
Social Networks	yes	Yes	Local	yes	No
Referral Networks	No	no	Local	no	Yes
Distributed Model	No	no	Local	no	Yes
Ad-hoc networks	No	no	Local	no	Yes

Table 4.4 Comparing PTEI with other Trust Models

4.7 Discussion

Reducing the perception of risk involved in trading with unknown e-commerce entities requires a new trust promoting framework. While cryptographic schemes can assure the integrity of the data received and the identity of a peer entity, they cannot by themselves guarantee the trustworthiness of entities for specific types of transactions. In the past e-commerce intermediaries such as payment gateways have played a major role in promoting trust in financial transactions. The right type of intermediaries can be selected automatically if they are classified on the basis of transaction categories they are authorised to endorse. Category specific endorsements were also used as a means of promoting trust in traditional commerce, together with experiential trust and trust based on recommendations. Creating an e-commerce trust framework combining such mechanisms poses many challenges. The trust network must be centralised if endorsement intermediaries are to be selected at runtime. Furthermore, it must be self-regulating if trust relationships are to reflect past direct experience and recommendations based on the experience of trusted neighbours. Meeting these challenges requires devising a category specific trust network, policies that make the network self-regulating, and efficient trusted paths generation techniques.

4.7.1 Institutional Trust

Reputation mechanisms through product ratings are used as a means of promoting trust in web sites such as ebay and Amazon. These ratings, which combine previous user experiences in the form of numerical value or written comments, allow other product users to make an informed decision. Social networking sites such as LinkedIn allow one to endorse others for a specific category of skills. However, the validity of such ratings relies on the rating and authority of raters themselves. The endorsement trust model presented in PTEI provides a stronger form of indemnity against trading risks by combining existing trust relationships with endorsements done by authorised intermediaries [167]. The level of indemnity is modelled by path endorsement trust (*PET*), which aggregates the existing trust relationships between endorsement intermediaries along the message path. The trust relationships themselves are a cumulative measure of past experiences. Trust relationships may be lowered or severed if trust breaches are detected along the message path. Endorsement capabilities, too, may be revoked if invalid endorsements are detected. Avoiding invalid endorsements and poor alliances is therefore in the self-interest of intermediaries that aim to maximise their endorsement incomes. Simulation results show that economic incentive/disincentive schemes are effective in making the trust relationships of more trustworthy intermediaries improve at a faster rate. Detecting trust breaches and invalid endorsements, however, is predicated on finding end-to-end security mechanisms that make each endorsement intermediary along the trusted path accountable. Such mechanisms are presented in chapter 5.

4.7.2 Self-Regulating Trust Networks

Past attempts at trusted path generation used recommendations and referrals from intermediaries to find the right services [40, 128, 133, 157]. The PTEI approach however, requires finding trusted endorsement intermediaries that can inspect and endorse key data based on experiential trust. New entities would initially require long endorsement chains as they may have very few established trust relationships. Costs and bandwidth for these long endorsements would severely limit their trading opportunities. To circumvent this problem, PTEI allows formations of new trust relationships in addition to evolution of existing ones. New trust relationships formed are initially assigned a low value reflecting their unfamiliarity. Trust evolution policies allow trust relationships to grow over time. Trust relationships are devalued for security breaches, failure of trading entities or endorsement failures.

McKnight and others [113] have proposed multidimensional trust models that includes institutional trust, trust disposition and trusting beliefs. In PTEI, endorsements which provide a form of institutional trust are combined with trusting beliefs about various domains and categories to update trust relationships. New trust relationships are established when indirect trust relationships exceed the trust transfer threshold value specified (which reflects individual trust disposition). Simulation results also suggest that trust transfer threshold should be allowed to vary over time if entities are to build and preserve their trust capital. Thus, PTEI policies allow criteria for establishing new relationships to be made more stringent as the number and extent of trust relationships grow. This models the common perception that "new traders often take higher risks when they have little to lose by trading with less well known entities". Thus, the growth of trust relationships in traditional commerce can be modelled more closely by combining trust evolution with trust transfer policies.

4.7.3 Response Time for Trusted Path Generation

Evaluating trust values for social networks and web services are highly time consuming making it difficult to find trustworthy participants or services at runtime [17, 133]. One novelty of PTEI is synthesis of trusted paths at runtime based on a centralized trust network that treats trust itself as a derived value. The extent of trust relationships are computed based on past experiences (both positive and negative) and trust dispositions of entities and intermediaries in different domains and categories. Furthermore, in an optimistic scenario, every transaction through the trusted path can be considered a success unless the centralized server is alerted of any trust breaches or endorsement failures. Any failure along a trusted path causes trust relationships to be adjusted. Unlike PTEI, many of the past trust prediction models use a distributed architecture [38, 40, 128], where trustworthiness is the aggregate of values obtained from a number of sources. Such techniques are inherently slow because of the communication delay involved. Moreover, trustworthiness may not be reliable as the node information from intermediaries themselves may not be reliable or up to date.

Two techniques devised in PTEI helped to reduce the response time even further. The first technique clustered intermediaries into hierarchical domains with domain names as part of entity identity. This allowed the solution search space to be restricted to only those clusters where either the sender, recipients or their intermediaries are located. Furthermore, searches can be carried out concurrently as domain networks are maintained separately. Using such a technique, trusted paths were generated within 5 milliseconds for groups of 200 entities with medium coupling (defined in Section 4.6.2). Based on complexity analysis presented in 4.5.3.4 for hierarchical networks, the elapsed time for generating a trusted path to 2 destinations in a network of 800,000 (200x200x200) entities will be 60 ms (2x3x2x5). The second technique created trusted paths offline, where the retrieval times are in the order of 0.25 ms for up to 25 recipients.

4.7.4 Security and Cost for Multiple Endorsements

Multiple endorsements are necessary when trading entities do not share common trusted intermediaries. With hierarchical intermediaries, endorsement at one level may be deferred until endorsement is obtained at another level. When multiple endorsements are needed, actions must be carried out in a predefined order with access to confidential data elements restricted to specific intermediaries. End-to-end schemes such as those presented in chapter 5 may be used to enforce such secrecy requirements. To reduce the cost of multiple endorsements, PTEI share common endorsements when data are sent to multiple recipients. Moreover, PTEI only requires endorsements when key messages are sent between trading entities.

4.7.5 Future Work

In this framework trust is treated as a derived value based on past experience, trusting beliefs and trust disposition; the model could be extended to include other factors that may influence trust such as the integrity and benevolence of a trustee. Experiential trust modelled in PTEI does not consider the value of transactions; rewards and penalties based on conduct can be made a function of transaction value. Distrust between entities can be considered when selecting trusted paths as distrust plays a major role in traditional commerce. The current model does not include implementation details of endorsement functions. More work must be carried out to define endorsement interfaces, quality of service, maintenance and access control. Trust transfer in the proposed model considers trust relationships of immediate neighbours only; it can be extended to allow greater trust transitivity. Finally the simulation results must be corroborated using real data in areas where multilevel endorsements are vital such as those used in news feeds, outsourcing, international contracts and online gaming services.

4.8 Conclusion

In this Chapter, an institutional trust framework was proposed to promote trust between unknown e-commerce entities. In particular, the following contributions were made.

- A new institutional trust framework was devised to help overcome the perception of risk involved in trading with unknown e-commerce entities. Inspired by trust establishment methods in traditional commerce, this framework allows key messages to be passed through trusted, category specific endorsement intermediaries. The endorsement intermediary network consists of nodes representing trading entities and authorised intermediaries, while edges represent the extent of trust relationships that exist between them. Path endorsement trust gives a measure of indemnity, based on the number of endorsement intermediaries and the trust relationships that exist between them.

- The endorsement trust network was made self-regulating combining direct and indirect endorsement trust. Trust evolution policies allow trust relationships with reliable intermediaries to grow over time. Trust transfer policies allow new trust relationships to be established, thus reducing the transitive depth between trading entities. Simulation results show these policies cause trust relationships with reliable entities to grow over time and that trust transfer policies are effective in modelling trust disposition. Hence, trust transfer and trust evolution policies can be combined to model trust in traditional commerce more closely.

- Trust was treated as a derived value based on past experiences and trust dispositions of entities and intermediaries. Large networks were decomposed into hierarchical domains using a clustering technique devised. Simulation results show that for each domain with medium trust coupling, the average search time for a trusted path grows linearly. Hierarchical organisation allows trusted paths to be retrieved within one second even for trust networks of up to 800,000 nodes. This performance can be further improved by generating domain trust trees offline.

Chapter 5: Security Schemes for Endorsement Services (SSES)

5.1 Introduction

Chapter 4 presented a trust model that can help alleviate distrust between unknown e-commerce entities by selecting intermediaries based on endorsement capabilities and trust relationships that reflect past experience. Maintaining relationships however, requires accountability schemes that can detect trust breaches. An accountability scheme allows every action to be associated with a specific entity [30]. By incorporating explicit cryptographic evidence it allows any misconduct to be proved to a third party adjudicator, similar to audit trails used in traditional commerce. Currently there are no standard accountability schemes for data sent to multiple recipients through common intermediaries. Such end-to-end schemes must be dynamically generated as the number and type of intermediaries vary with the transaction type and trust requirements of interacting entities. The proposed framework allows such end-to-end schemes to be derived by combining proven two-party schemes.

End-to-end security schemes can detect and prove trust breaches, but not prevent them. Preventing trust breaches requires combining end-to-end schemes with techniques for selecting reliable intermediaries. Reliable intermediaries with built–up trust capital are less inclined to misbehave when it is known that trust breaches will lead to a loss of trust relationships. This interdependency between trust relationships and end-to-end security is captured in this framework in two ways. Firstly, minimum trust relationships can be specified when selecting message paths, thus lowering the likelihood of trust breaches. Secondly, when an entity trust breach is detected using end-to-end accountability schemes, trust relationships with that entity are lowered or severed depending on the type of trust breach. This allows existing trust relationships to reflect past conduct.

Although end-to-end protocols were created for specific applications in the past [89, 144], little or no work has been done for endorsement intermediaries. When endorsement intermediaries are used, the data originator and recipients may want an end-to-end assurance about data and successive endorsements along the path. Furthermore, a higher level endorsement or an indirect endorsement may not be obtained unless all previous endorsements have been carried out in the right order. The proposed framework provides such an assurance in the form of a certified trusted path through intermediaries with required endorsement capabilities.

The security challenges in endorsement security are similar to that faced in content-transforming intermediaries, which include data corruption, eavesdropping, false responses by adversaries and repudiation of message despatch or receipt [144, 145]. However, the excessive cost of cryptographic operations required in each intermediary has caused data security to be neglected [144-146]. In the past others have attempted to reduce performance overheads by exploiting any inherent parallelism in

data relationships [146]. In this framework messages sent to multiple recipients are made to share common endorsements, thus reducing intermediary overheads. The use of fine-grained security properties and schemes enforcing them help improve security performance trade-offs by providing the right level of security. The right level of security along each edge of the trusted path is determined as a function of security requirements by data originator and all recipients using that edge.

5.1.1 Main Contribution

Before e-commerce messages can be sent to multiple recipients through endorsement intermediaries a number of research challenges have to be overcome. Firstly, security must be provided for both data and endorsements. Secondly, security schemes must incorporate the necessary evidence to make all intermediaries accountable for their actions. Finally, end-to-end security schemes must be created dynamically, as the number of intermediaries and the type of security properties may vary. This section describes the main contributions made in meeting these challenges.

- The notion of proof obligations introduced facilitates reasoning about accountability for various security properties. End-to-end accountability schemes for security properties providing assurances to recipients (A,DI,TB) are derived by transferring proof obligations through intermediaries until they can be discharged using direct evidence from the data originator. Similarly, end-to-end accountability schemes for security properties providing assurances to data source (RNR) are derived by delegating proof obligations through intermediaries until direct evidence is obtained from all recipients. End-to-end assurances follow naturally when all proof obligations are discharged either directly or indirectly. Conversely, any trust breach or invalid transaction can be narrowed to a specific intermediary or trading entity failing to discharge its proof obligations. Past research with accountability was mainly restricted to verifying whether existing protocols can make participants accountable for their actions. E-commerce protocols must be designed with accountability as a core requirement if legal recourse is to be provided when intermediaries fail to discharge their obligations.

- An SSES scheme devised allows server signed metadata consisting of data hash, data category, trusted path and required security levels along the path to provide a form of protocol description. Each intermediary and recipient along the trusted path enforces the security levels specified using the proven two party schemes in their possession. Such a scheme allows end-to-end security and trust to be enforced.

5.2 Statement of the Problem

Generating provably secure end-to-end security protocols through endorsement intermediaries pose a number of challenges that must be overcome. The main ones are highlighted below.

- End-to-end schemes for messages sent through intermediaries must provide security assurances for both the data and all intermediary endorsements. The validity of final endorsement is dependent on each endorsement along the path being valid for a given data category and data.
- Accountability requires entities and intermediaries to discharge their proof obligations by producing the necessary cryptographic evidence from the message source or message recipients. However, direct evidence from the source or destination may not be possible as messages may be sent through multiple intermediaries. Hence, mechanisms must be devised to allow entities and intermediaries to discharge their proof obligations indirectly through others.
- Accountability requires explicit cryptographic evidence that can be presented to a third-party arbitrator. For example, the nonces used for verifying recency in Chapter 3 cannot be presented as explicit evidence of recency to a third-party arbitrator as these nonces were locally generated.
- The security level along the common trusted path must meet the security requirements of all recipients reached through them. Furthermore, the cryptographic schemes for these security levels must be sequentially composable (defined in 2.2.2.2). For example, if security properties *{A,RNR}* are used when sending message *M* from *P* to *I*, and *{A}* when sending *M* from *I* to *R*, and *{RNR}* when sending *M* from *I* to *S*, then the schemes used by *{A,RNR}* and *{A}* for authentication and *{A,RNR}* and *{RNR}* for non-repudiation must be sequentially composable.

5.2.1 Research Questions

- Validity of a message sent through multiple endorsement intermediaries depends on integrity of data and endorsements along the path. This leads to the research question: How can end-to-end schemes be devised that provide both data and endorsement security?
- When messages are sent through multiple endorsement intermediaries, misconduct by any entity or intermediary may cause a transaction failure. This leads to the research question: How can end-to-end schemes be devised that make all endorsement intermediaries and trading entities accountable for their actions?
- Protocols must be synthesised at run time if protocol paths through intermediaries are to reflect existing trust relationships. This raises the research question: How can newly synthesised protocols be distributed and enforced by all entities and intermediaries along the protocol path?

5.3 Outline of the Solution

The end-to-end schemes devised make all entities and intermediaries accountable for their actions. The notion of proof obligation and the means for discharging, delegating or transferring them provides an elegant mechanism to derive end-to-end security properties, as outlined in Section 5.3.1. Data sent through intermediaries include data from the source and endorsements by intermediaries. Endorsements provide the necessary evidence to each subsequent intermediary along the trusted path, as outlined in Section 5.3.2. All messages are accompanied by metadata signed by the trust server, which includes both the trusted path and the minimum security levels along the path, as outlined in Sections 5.3.3 and 5.3.4. The metadata signed by the trust server serves as the certified protocol description, allowing protocols to be synthesised and executed dynamically. The cryptographic schemes are revised to incorporate explicit cryptographic evidences to make all intermediaries and entities accountable, as outlined in Section 5.3.5.

5.3.1 Deriving End-to-End Security Schemes

End-to-end schemes for data sent through intermediaries are derived by combining proof obligations (liabilities) described using accountability logic (presented in Section 2.5) with attestable evidences for security properties derived in Chapter 3. These attestable evidences can be used for transferring, delegating or discharging such obligations. Proof obligations can be discharged directly when direct evidence is obtained from source or recipients. When direct evidence cannot be obtained from all recipients proof obligation can be delegated. Similarly when direct evidence cannot be obtained from source it can be transferred. Protocol entities lacking the necessary evidences to discharge, delegate or transfer their proof obligations are considered to have breached trust assumptions. The notion of proof obligation is first presented using two examples in this section, before being formalized in Section 5.5.5.

Example: Informal Notion of Destination Proof Obligation

An employer may use the recommendation of an authorised agency *A* as the basis for hiring a candidate, which may be based on direct knowledge of the candidate or an earlier recommendation by another authorised agency *B*. In the second scenario, the employer may hold the agency *A* liable for the hire while *A* may hold the other agency *B* liable. This process may be repeated transitively where each agency is obliged to prove it has direct knowledge of the candidate or has received an earlier recommendation. In SSES, destination proof obligation models a similar notion for end-to-end security properties {A, DI, TB} where the recipient and intermediary may transfer liability to its predecessor until it can be discharged directly.

Example: Informal Notion of Source Proof Obligation

Assume a CEO of a company is legally bound to inform each employee about a stock option and to gather evidence of its receipt. The CEO may choose to carry it out by informing all subordinates and getting an undertaking that they will delegate it recursively until all employees are informed. In addition the CEO may attach an endorsed organizational structure to make the reporting structure explicit. Also, assume the company rules require anyone failing to get evidence of delegation to inform the secretary (allowing other arrangements to be made). In such a case, the CEO has discharged his or her liability as long as he or she can show evidence of informing and delegating to all subordinates. In SSES a source proof obligation models a similar notion which can be used by the data originator and subsequent intermediaries along the tree to discharge their liabilities for properties such as {RNR}. Unlike destination proof obligations, source proof obligations require evidence from all successors as SSES allows data to be despatched to multiple recipients through common intermediaries.

5.3.2 Schemes Devised for Endorsement Security

Messages through endorsement intermediaries may consist of earlier endorsement, in addition to data from source and server signed digest. The variable part contains the endorsement from the predecessor in the trusted path and forms the basis for subsequent endorsement. The first endorser along any trusted path must possess direct knowledge of the originator. Similarly the last endorser must possess direct knowledge of the recipient.

5.3.3 Technique for Enforcing Path Security

Each cryptographic scheme in the proposed framework named Security Schemes for Endorsement Services (SSES) includes a server signed digest made up of the valid trusted path, a data category and a hash of the original data. The elements in the server signed digest allow data and the node through which they are received to be validated before discharging, delegating or transferring proof obligations.

5.3.4 Enforcing Minimum Security Levels along a Message Path

The minimum security level between any two adjacent entities in a message path is expressed as a function of the security requirements of all recipients using that path. The trust server specifies the minimum security level required along every edge in the trusted path by annotating security levels in the trusted path as in [*A*[*B:8*[*C*][*D:2*]][*E:11*[*F*][*G:2*]]], where the letters represent entities and intermediaries along the path and the numbers represent the security levels along the path. Fine-grained schemes are created by combining basic schemes, similar to the approach used in Chapter 3. These pre-created schemes allow an entity or intermediary to select the scheme matching the security level specified along the path.

5.3.5 Revised Schemes with Explicit Cryptographic Evidence

The time-bound property combines predecessor signed despatch-time with expiry time in the server signed digest to enforce the time-bound property. Any intermediary can show that data was despatched with time-bound property if the difference between its own despatch time and that of its predecessor is less than the expiry time. The expiry time specifies the m9aximum delay between nodes, which includes both the communication delay and the processing time. A time-bound can be placed as the number of intermediaries in the trusted path is limited by the maximum transitive depth (*MTD*) allowed. It is assumed all trusted intermediaries have synchronised clocks.

5.4 Literature Review

This section outlines the research relevant to end-to-end security and content transforming intermediaries. The initial section outlines research problems in e-commerce end-to-end security. Research issues in accountability and delegation relevant to end-to-end protocols are presented next.

5.4.1 Security Threats of Content Transformation Intermediaries

Intermediaries are software entities that intervene in the flow of information from message originators to recipients. Their main aim is to support seamless access to services for different types of clients by providing functions like customisation, annotation, filtering and transcoding [147]. The transformation may include format translation or other forms of filtering such as virus removal or watermarking [146]. Annotations may include some form of endorsement or additional information that may be used by subsequent intermediaries or recipients. Content transformation through intermediaries introduces many security hazards, including unauthorised access and alterations [145]. Valid content-transforming intermediaries themselves may pose threats when multiple intermediaries are involved [148]. Past schemes to enforce content integrity include use of metadata expressing modification policies. However, the increased overheads resulting from these schemes limit their applicability, especially for large data volumes [146].

Some content transforming systems allow alteration of signed data by trusted intermediaries by sharing the necessary keys [144]. These schemes, however allow intermediaries to make random changes to the original data which may result in invalid information being passed to recipients. Furthermore, sharing keys with multiple intermediaries makes it impossible to attribute changes to any one entity. Although content transformation techniques have been an active area of research, relatively little attention has been paid to data security until recently. The size of data involved and the number of intermediaries pose many challenges related to security and performance overheads. Reducing intermediary overheads requires minimising the number of intermediaries, especially when multiple recipients are involved.

5.4.2 Protocols for Intermediary Communication

Trusted intermediaries play a vital role in e-commerce by providing services such as anonymity preservation and quality of service guarantees [28]. Such services require additional transformation and annotations of data flowing from originator to recipients. Intermediary responses may consist of either additional information for interpreting data from message origin, guarantees, endorsements or caveats [29]. Currently, intermediaries communicate between themselves and with servers by using standard protocols such as HTTP, SMTP and RMI. Moving towards an open distributed setting poses many research challenges, including the need for flexible mechanisms to allow richer interaction between other intermediary components, message originators and diverse clients. Also, flexible

mechanisms are needed for authenticating third parties requesting services, and for enforcing trust policies at the granularity of data elements [29]. When standard point-to-point security protocols such as SSL, are used with intermediaries, multiple connections are needed to enforce end-to-end security. Furthermore, SSL does not allow secrecy requirements to be specified at the granularity of message element, which is required to restrict access to specific intermediaries. These limitations make it necessary to devise end-to-end schemes that enforce the right level of security, including secrecy requirements at data element level.

5.4.3 Need for Fine-grained End-to-End Schemes

In the past, security properties were often combined to create stronger properties to meet specific needs. The recency property, guaranteeing freshness of messages, is often combined with the authentication property to create a stronger form of authentication [33]. However, the use of the recency property based on nonces (as presented in Chapter 3) has limited applicability as it does not allow claims about freshness to be proved to third parties [30]. Many security relevant attributes such as keys expire after a specific time. The period of validity for a message using those keys can be made explicit by including a time-bound [149]. Using a time-bound can also avoid stale messages being despatched through intermediaries. A time-bound property can be combined with other properties to create stronger versions of authentication and non-repudiation [48]. An arbitrator can verify whether the message sent is valid at a particular point in time, if it has an explicit reference to time of message despatch.

An organisation responding to a tender expiring soon may want to combine the time-bound property with non-repudiation. Intermediaries that are slow to respond can be easily identified if the time-bound property also includes time of message despatch. Unlike other security properties, the onus of proof for the non-repudiation property is placed on the message originator. The receiver non-repudiation protocol is considered fair if neither sender nor receiver gets an advantage by terminating prematurely. Therefore, fair non-repudiation schemes must ensure that the receiver gets the data and the sender gets evidence of receipt, or neither party can get any useful information [105]. Most fair exchange non-repudiation protocols place a time interval within which non-repudiation evidence must be presented [150]. The period of validity can be made explicit by combining the non-repudiation and time-bound properties. The lack of end-to-end schemes also poses a major challenge in web services where all intermediaries may not be trusted to the same extent. Existing specifications such as WS-Security provide only pair-wise assurances [2].

5.4.4 Accountability in E-commerce

Accountability is defined as the property whereby association of a unique originator with an action or object can be proved to a third party [30]. Commercial transactions must have the built-in mechanism to hold peer entities accountable for their actions. Proving an assertion about an action differs from believing that assertion to be true. When believing an assertion, an entity becomes convinced that it is

true, though it may lack the explicit evidence to prove it to an arbitrator. Proofs have been classified as strong proofs if the evidence can convince any entity about the validity of an assertion, and weak proofs if they can only convince specific entities [30]. Protocols can be analysed for accountability, based on assumptions such as "private keys are not divulged by honest entities", and "signatures can be associated with unique entities". The accountability proofs for protocols have been derived from generic properties and properties applicable only to systems using digital signatures [30]. The initial assumptions assert the origin of messages signed by private keys and associate specific meanings to messages. This approach provides a means to analyse the accountability of existing protocols, and resolve disputes by third parties. It also provides a systematic way of identifying the required set of protocol messages to match the transaction goals, thereby identifying redundant messages in existing protocols. However, these techniques are useful only for analysing existing protocols.

5.4.4.1 Accountability Delegation

Security protocols created on the fly should configure trusted paths dynamically, based on the type of transaction, the threat posed by the environment and the relationship between entities [115, 151, 152]. If disputes are to be resolved by third parties, interacting entities with no prior relationships must delegate accountability to trusted intermediaries [153]. Most of the delegation protocols designed in the past allowed only delegation of rights. Delegation of accountability allows one entity to rely on another to enforce the necessary security properties. However, a delegated entity cannot be held accountable unless its explicit consent to enforce the required property is proven.

5.4.5 Summary and Evaluation

Content transformation intermediaries have been an active area of research reflecting the need for various forms of services including filtering, transcoding and watermarking. Such systems require messages intended for one or more recipients to be passed through intermediaries with varying access rights. Path security is also vital, when guarantees are needed about the order in which services are carried out. For time-critical applications it may also be necessary to enforce time bounds. However, problems related to data security have not been addressed adequately in such systems [144-146]. Accountability is also vital for intermediaries which influence e-commerce decisions, such as whether to collaborate with an unknown entity. However, a search of the literature review failed to reveal any past technique that makes such intermediaries accountable.

The proposed solution makes intermediaries selected at runtime accountable by deriving end-to-end schemes dynamically. Secrecy schemes allow data access to be restricted to specific entities. Path security is enforced by sending data together with server signed metadata. Use of common endorsements for data sent to multiple recipients helps to reduce security and service overheads.

5.5 SSES Model Elements

This section describes how the main SSES elements combine to provide end-to-end security for messages sent through endorsement intermediaries. Section 5.5.1 highlights some limitations of the interleaved schemes as devised in Chapter 3. Section 5.5.2 presents an endorsement chain made up of a series of endorsements along a trusted path. Section 5.5.3 defines end-to-end properties. Section 5.5.4 describes the structure of server signed trusted paths included in digests. Section 5.5.5 outlines how proof obligations are used for creating end-to-end schemes. Section 5.5.6 presents security schemes devised for data and endorsement chains. Section 5.5.7 describes the complementary roles played by dynamic trust relationships and end-to-end schemes.

5.5.1 Comparison with Interleaved Schemes Devised in Chapter 3

In Chapter 3, two-party cryptographic schemes for security properties were interleaved to create schemes for any number of recipients forming a sequence. By piggybacking messages and forming a cycle these interleaved schemes helped to reduce both the traffic at the source node and the number of message hops in the protocol. However, the interleaving technique assumes intermediaries are passive, with no message-related endorsements or responses, while piggybacking increases bandwidth through additional payloads. Any delay in one of the entities impacts the all other entities relying on piggybacked messages, thus adversely affecting end-to-end response time. Also, piggybacking is applicable only when entities lie in a linear path. Interleaved schemes also assume that all message originators know the public keys of all recipients, and all recipients know the public keys of message originators, thus limiting their applicability. The revised scheme presented in this chapter allows more flexible path configurations for intermediaries where security levels along a message path are allowed to vary, reflecting the security requirements of the originator and recipients. Entities and intermediaries are required to maintain the public keys of only those entities/intermediaries they trust.

In SSES, the endorsement function (*f*) returns a precise value based on data, data category previous endorsement (if any) and the endorser, based on existing rules or laws. The endorsement value can be compared to the endorsement made by a certified notary for a specific type of document based on existing laws in that region.

DEFINITION 5.1 INTERMEDIARY ENDORSEMENT FUNCTION (F)

Let *EI be the set of all endorsement intermediaries,*
DC be the set of all data categories,
D be the set of all data,
EV be the set of all endorsement values,

then $F : EI \times DC \times D \times EV \rightarrow EV$ *is the endorsement function that returns an endorsement value based on endorsement intermediary, data category, data and previous endorsement. Note when the previous endorsement is set to null a direct endorsement is performed.*

An endorser may perform different functions based on the data category (*DC*). An endorsement intermediary indirectly determines the domain in which endorsement is performed and whether it is direct endorser, indirect endorser or either. Any invalid endorsement value may lead to loss of loss of endorsement capability if proven.

TERMINOLOGY 5.1 TRUST SERVER

In SSES the trust server in addition to maintaining trust networks and forming trusted paths also signs digests containing hash of data, data category and the trusted path thus acting as a protocol server.

5.5.2 Endorsement Chain along Trusted Path

The trusted paths presented in Chapter 4 take into consideration the hierarchical organisation of endorsement intermediaries (common in the physical world), where endorsement at one level may depend on endorsements at all previous levels. Therefore, SSES schemes are designed to provide security for both the source data and the endorsement chain. Each endorsement value is expected to comply with the output from the intermediary endorsement function (*f*) taking as input fixed data *d* (from origin), previous endorsement dv_i, data category *DC* and the endorser. For example, in Figure 5.1 the data *d* originating in *O* is passed through endorsement intermediaries E_1, E_2 and E_3. E_2 forwards the fixed part *d* received without any alteration whereas the variable part dv_2 is given by $dv_2 = f(E_2, DC, d, dv_1)$. The values dv_0, dv_1, dv_2 ... dv_n form the endorsement chain and *sp* refers to the security property. The term dv_0, set to a null value, is used only for the purpose of consistency of notation.

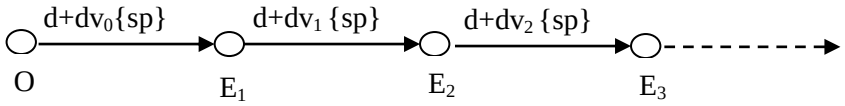


Figure 5.1 Intermediary Messages with Fixed and Variable Parts

5.5.3 End-to-End Security Properties

Past attempts to secure data sent through content transforming intermediaries included schemes for standard security properties including data authentication, data integrity and data non-repudiation [2, 145]. SSES must provide these such assurances for both data elements from the source and endorsements made by intermediaries. Path security is also essential when data paths are restricted [144]. SSES enforces path security by ensuring that the order of intermediary endorsements for data matches the trusted path for data included in the trust server signed digest. In addition, accountability is enforced in SSES by incorporating evidences that can be archived and presented to third parties when necessary. The list of end-to-end properties that incorporate path-security and accountability are listed in Table 5.1.

E-commerce transactions can be secured if accountability is used as a fundamental property [153]. Recency, one of the basic properties in Chapter 3 and its underlying scheme presented does not allow for accountability. For example, an entity *B* can convince itself that the data received from *A* is recent, if *A* also sends a signed digest made up of a hash of the data together with a nonce *A* sent recently. However, B cannot present this digest as proof of recency to a third party, as the uniqueness of the nonce cannot be proved. Hence, SSES uses the time-bound property (*TB*) instead which can be enforced with explicit evidence. Note all the security properties enforced at message level ({*A,TB,RNR,NI*}) are required to guarantee that each endorsement along the path is valid for given data, data category and previous endorsement. A valid endorsement is one that adheres to the rules associated with the intermediary endorsement function assigned (Definition 5.1).

Symbol	**Property**	**Meaning**
A	Path Authentication	Each recipient can prove that the chain of data and endorsements are directed to the correct intermediary/entity along the trusted path and each endorsement along the path is valid for given data, data category and previous endorsement (if any).
TB	Path Time-Bound	Each recipient can prove it has received data and endorsement chain within the time-bound allowed by the trust server and each endorsement along the path is valid for given data, data category and previous endorsement (if any).
RNR	Path Non-Repudiation	A message-originator can prove that each entity or intermediary along the trusted path cannot deny receiving a chain of data and endorsement where each endorsement is valid for given data, data category and previous endorsement (if any).
DI	Path Data Integrity	Each recipient can prove that the chain of data and endorsement received through a trusted path has not been tampered with along the way and each endorsement along the path is valid for given data, data category and previous endorsement (if any).
FS	Secrecy	Fine-grained secrecy at message element level.

Table 5.1 End-to-End Security Properties

Distinct schemes are needed for the various combinations of security properties, as listed in Table 5.2 to enable fine-grained end-to-end security. For example, by combining the properties *TB* and *RNR* the message originator can prove that all recipients were in possession of the messages before the message

expiry time. SSES assigns an index (*SPI*) to each of the 15 different combinations of security properties as shown in the second row of Table 5.2. These indices (*SPI*) are used to specify the security level along the trusted path. The secrecy property — being a non-correspondence property is not shown in Table 5.2; it uses schemes that are independent of others, allowing it to be combined with any of the other schemes.

A	DI	RNR	TB	A,DI	A,RNR	A,TB	DI,RNR	DI,TB	RNR,TB	A,DI,RNR	A,DI,TB	A,RNR, TB	DI,RNR, TB	A,DI, RNR,TB
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Table 5.2 Security Properties and their Indices

5.5.4 Example of Server Signed Trusted Path

In PTEI framework presented in Chapter 4, trusted paths between data originator and recipients were represented in the form of spanning trees. Trusted paths were generated on the basis of minimum trust relationship and maximum transitive depth (refer to Terminology 4.1) specified. SSES allows all recipients and intermediaries to verify data are passed along valid trusted paths signed by the trust server. For example, assume that trading entity *A* wants to send a finance-related data *d* of category *Fin* (finance) to entities *C* and *D* with minimum trust relationship *n* and maximum transitive depth 1 (maximum number of intermediaries set to 1) with authentication assurance. Further, assume that *B* can be an intermediary from *A* to *C* and *D* for finance-related transactions, with all trust relationships exceeding *n*, as shown in Figure 5.2. In such a scenario, data *d* can be sent to *C* and *D* via *B* and the trusted path signed by trust server *TS* will take the tree structure shown below, represented textually as in [*A*[*B*[*C*][*D*]]]. In SSES all end-to-end schemes incorporate a certified trusted path made up of a server signed digest consisting of a security property index (*SPI*), data category (*DC*), valid trusted path from data origin [*A*[*B*[*C*][*D*]]] and a hash of data *h(d)* as in $\{SPI,DC,h(d),[A[B[C][D]]]\}_{priTS.}$

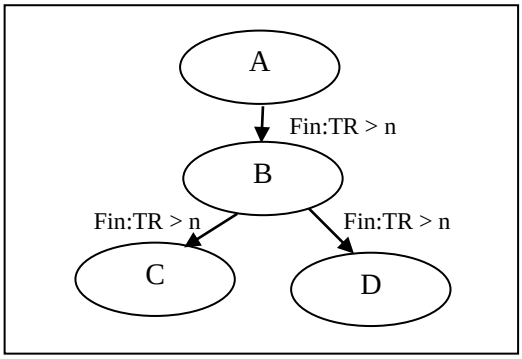


Figure 5.2 Trusted Path from Message Originator A to Recipients C and D

5.5.5 Enforcing Accountability through Proof Obligations

Much of the past work done on accountability has been limited to analysing existing protocols for accountability properties [30, 153]. Enforcing accountability for fine-grained end-to-end protocols requires a derivation-based approach, as the number of intermediaries and recipients in a trusted path may vary. The basic end-to-end security properties in SSES can be broadly classified into those giving

assurances to recipients and those giving assurances to message originators. Authentication, data integrity and time-bound properties assure recipients, while the non-repudiation property assures data-originators. In both cases end-to-end evidences cannot be sent or received directly making intermediary involvement inevitable. Thus, end-to-end assurances in SSES require data originators and recipients to combine authority granted by central trust server with evidences from that intermediary. These indirect evidences together allow any entity to indirectly discharge any liability either by transferring or discharging proof obligations.

TERMINOLOGY 5.2 PROOF OBLIGATION

Proof obligations (PO) are liabilities placed on entities and intermediaries that must be discharged using explicit cryptographic evidence. Proof obligations are further classified into destination proof obligations (DPO) and source proof obligations (SPO) reflecting where initial proof obligations are placed. Discharging destination proof obligations (DPO) require evidence from the data originator while discharging source proof obligations (SPO) require evidence from all recipients.

Note the SSES schemes allow data to be sent to multiple recipients to share common endorsements, thus reducing intermediary costs. A tree structure is used to represent all intermediaries and recipients. Discharging source proof obligations requires evidence from all subsequent branches or leaves in the tree, which represent intermediaries and recipient entities respectively. Proof obligations can be discharged only after evidence presented by other intermediaries or entities are validated by the trust server signed digest. The explicit cryptographic elements in the schemes can be presented as evidence to a third party, when necessary. An entity or intermediary unable to discharge a proof obligation based on direct evidence from the data originator or recipients may transfer or delegate it to other intermediaries using explicit cryptographic evidences from either predecessor or successors in the trusted path. Transferring destination proof obligation requires evidence from its predecessor, while delegating source proof obligations requires evidences from all successors. In Figure 5.3, the source proof obligation for {*RNR*} is placed on *E1*, while destination proof obligations for {*A*} and {*TB*} are placed on *E2* and *E3* respectively for data *d* is sent from E_1 *to* E_2 and *E*3 via the trusted path through intermediary I.

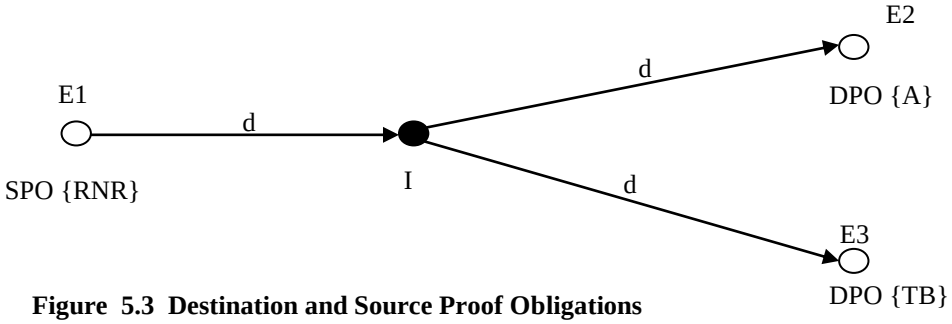


Figure 5.3 Destination and Source Proof Obligations

The Figure 5.4 shows how these proof obligations are either transferred or delegated through the intermediary I subject to receiving the necessary evidence, until all obligations are discharged. Note when multiple proof obligations are involved evidence along all edges are combined subject to non-interference.

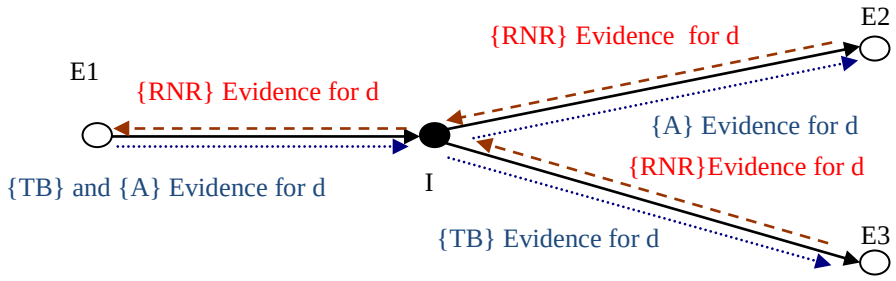


Figure 5.4 Evidences Required to Delegate/Transfer/Discharge Proof Obligations

If an intermediary presents insufficient evidence or incorrect evidence which cannot be validated, or fails to respond within the stipulated time, the entity terminates the end-to-end protocol midway and sends an exception message to the trust server. In response, the trust server downgrades the trust relationship along that path and attempts to find an alternative trusted path, thus avoiding the intermediary that caused the exception.

5.5.6 End-to-End Schemes for Data and Endorsement-Chain

One main challenge for end-to-end security is to provide the required security assurances for messages sent through intermediaries without restricting their actions [154]. Although content transforming intermediaries has been an active area of research in the recent past, issues related to content security were not addressed to the same extent [154]. SSES security schemes are designed to provide fine-grained end-to-end security for data elements from the entity data origin and for the chain of endorsements by intermediaries. Security for the chain of endorsements is necessary as it allows an intermediary to justify its own endorsement based on its predecessor, thus creating an audit trail.

As an example, consider the flow of data d_f of category DC along the trusted path from intermediary X to Y as shown in Figure 5.5. The data from X to Y, d_{XY} combines both the fixed part d_f and the variable part dv_{XY}. In other words, $d_{XY} = df \bullet dv_{XY}$ where $\bullet$ denotes the concatenation operator. The data is sent together with signed hash elements of fixed and variable parts. The hash of the fixed part $h(df)$ is signed by a trust server, while intermediary X signs the combined hashes of variable and fixed parts, denoted by $h(d_{XY}) = h(dv_{XY}) + h(df)$. The trust server signed digest allows data to be verified. The common hash $h(df)$ in the digest signed by X allows it to be linked with the trust server signed digest, while the hash of the variable part allows response (endorsement) from its predecessor in the trusted path to be verified. The intermediary Y then forms its own response dv_{YZ} based on df and dv_{XY} received securely from its predecessor along the path. The same scheme is then used to forward its own response and the fixed message part. Note that both server and predecessor signed elements may

contain other elements on top of the fixed data and the variable endorsement. Additional elements (such as trusted path) that are used for enforcing specific end-to-end security properties are presented in the next section.

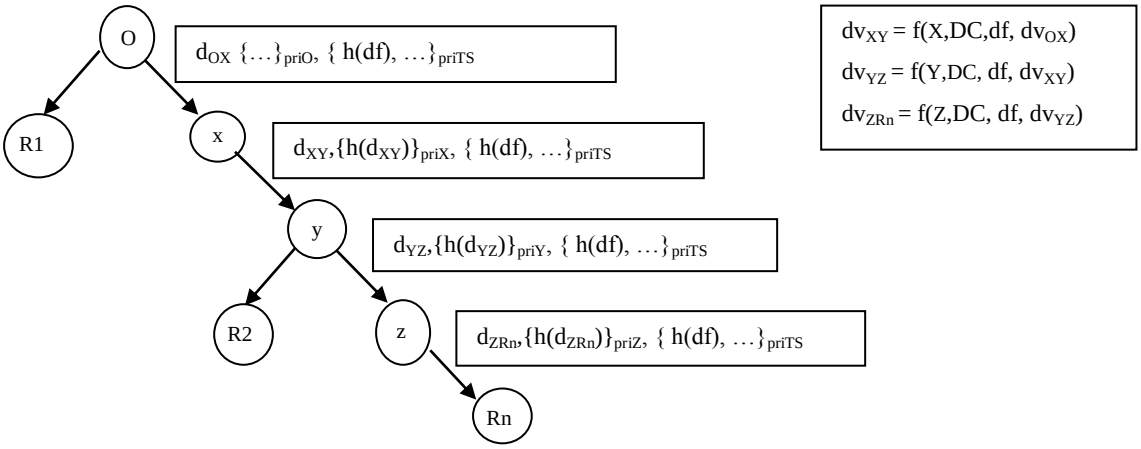


Figure 5.5 Verifiable Basis Path for Intermediary Generated Responses

All the SSES schemes described in the next section use this technique for enforcing end-to-end security for the fixed part.

5.5.7 Trust-Relationship and End-to-End Security

The PTEI techniques presented in Chapter 4 allow the trust server (*TS*) to determine a trusted path based on the trust requirements specified. The SSES accountability schemes presented in this chapter allow intermediaries breaching trust assumptions to be detected. Trust relationships with entities breaching trust are either lowered or severed, depending on domain trust policies. Hence, SSES and PTEI techniques have a circular relationship as they depend on each other. One major benefit of the proposed framework is that it allows interdependence between trust and security to be modelled explicitly.

5.6 SSES Model Details

This section presents the model details of SSES. Section 5.6.1 describes end-to-end security properties and when they may be prescribed. Section 5.6.2 identifies destination and source proof obligations associated with end-to-end security properties, which must be discharged, delegated or transferred. Section 5.6.3 summarises the role played by proof obligations. Section 5.6.4 presents examples of end-to-end schemes based on evidences needed for discharging proof obligations. Section 5.6.5 presents examples of composed end-to-end schemes derived by combining evidences needed for discharging multiple proof obligations.

5.6.1 End-to-End Security Properties and Proof Obligations

This section presents end-to-end security properties (defined in Table 5.1) in greater depth, including property-related proof obligations placed on message originators and recipients.

5.6.1.1 End-to-End Authentication Property

The end-to-end authentication property provides each recipient with the assurance that data and endorsements were sent authenticated from the message origin. It is enforced by requiring each recipient to discharge the destination proof obligation that data and a valid endorsement were despatched authenticated, along each edge of the trusted path from the data origin. The proof obligation can be discharged (indirectly) by transferring it to its predecessor in the trusted path (tree).

5.6.1.2 End-to-End Time-bound Property

The end-to-end time-bound property provides each recipient with the assurance that data and endorsements are timely. It is enforced by requiring each intermediary to discharge the destination proof obligation that the time duration between each subsequent despatch along the trusted path (starting from the source) is lower than the time interval permitted by the trust server. The proof obligation can be discharged (indirectly) by transferring it to its predecessor in the trusted path (tree).

5.6.1.3 End-to-End Receiver Non-Repudiation Property and Scheme

The end-to-end receiver non-repudiation property provides the message originator with non-repudiable evidence that receipt of data and endorsements cannot be disputed by any of the recipients. It is enforced by requiring the originator to discharge the source proof obligation that data and endorsements were sent along a trusted path and received by all recipients. Source proof obligations can be discharged indirectly by delegating it to all successors along the trusted path.

5.6.1.4 End-to-End Data Integrity Property

The end-to-end data integrity property provides recipients with non-repudiable evidence that data and endorsements were not tampered with. It is enforced by requiring each recipient along the trusted path to discharge the associated destination proof obligation. This proof obligation can be discharged indirectly by transferring it to its predecessor in the trusted path.

5.6.1.5 End-to-End Secrecy Property

End-to-end secrecy allows sharing of confidential information without disclosing it to intermediaries. A particular data item *m* is a secret at the end of the protocol run if an intruder or an entity that is not part of the secrecy group cannot obtain *m* during the run of the protocol [69]. Secret elements in SSES can only be accessed by entities that are part of a pre-established group. It is assumed that no member of a secrecy group will reveal its secret elements and all secret message elements in SSES are encrypted at the source using relevant group keys. Thus, any message containing secret elements can be passed safely through entities that are not part of the secrecy group. To ensure no secrets are leaked indirectly, secrecy groups of derived elements are made a subset of secrecy group of base elements (as in Chapter 3).

5.6.2 Discharging Proof Obligations

Proof obligations for basic security properties specify the cryptographic evidence needed to discharge them directly, or indirectly by transferring or delegating them. SSES techniques allow end-to-end schemes to be created by combining such cryptographic evidences. In addition each intermediary is required to prove its endorsement for specified data, data category and previous endorsement matches the value returned by the intermediary endorsement function (Definition 5.1). For all end-to-end security properties, discharging, transferring or delegating associated proof obligations can be done only after validating evidence presented by entities with the evidence from the trust server. If an anomaly is detected while the protocol is still in progress, an exception message is sent to the trust server, allowing corrective action. If an anomaly is detected afterwards, the archived cryptographic information from various entities can be combined to detect the entity or intermediary causing the problem. In either case, the trust server may lower or sever the trust relationship when an entity or intermediary is noncompliant.

The Logical Framework

The logical framework for proving end-to-end schemes through endorsement intermediaries combines SPCL used in Chapter 3 with accountability logic presented in Section 2. While SPCL allows reasoning about security schemes and provides the basis for combining them securely, the implicit evidence used in such schemes cannot be presented to a third party to make an intermediary accountable. The inference rules in accountability logic allows cryptographic evidences needed for transferring, delegating and discharging proof obligations to be specified explicitly. All the proofs in this Section make the perfect encryption assumption stated in Section 2.1.2.

TERMINOLOGY 5.3 PRIMITIVE FORMULAS AND SEMANTICS

- *dig* refers to a digest and *dig.x, dig.y* refer to elements *x* and *y* contained in that digest.
- E_i,E_{i-1} refer to intermediaries or entities
- $E_i pri$ refers to private key *of* E_i
- *h(d)* refers to hash of data *d*

- *TD, TA* and *TE* refer to time of despatch time of arrival and expiry time.
- *Tree* refers to the tree representing the trusted path from message origin *O*
- *Sub-tree(N)* refers to the sub-tree of *Tree* representing rooted in node *N*
- *Next(Tree,X)* returns the successors of *X* in *Tree*
- *sp* represents a security property variable representing one of {A,DI,TB,RNR}
- The symbol ● is used to indicate concatenation of terms.

DEFINITION 5.2 DESTINATION PROOF OBLIGATIONS

Let *d be the data,*
E_n be entity or intermediary along the trusted path
dv_n be endorsement received by entity E_n
DC be the category of data
O be the entity of data origin
$sp \in \{A,DI,TB\}$

Then the destination proof obligation DPO[$d+dv_n$,O,DC,E_n,sp] states that E_n is obliged to prove that both the data d originating in O, and endorsement dv_n from its immediate predecessor are received with security assurance sp.

DEFINITION 5.3 SOURCE PROOF OBLIGATIONS

Let *d be the data,*
E_n be entity or intermediary along the trusted path
dv_n be endorsement received by entity E_n
DC be the category of data
O be the entity of data origin
$sp \in \{EA, RNR\}$

Then the source proof obligation SPO[$d+dv_n$,O,DC,E_n,sp] states E_n is obliged to prove all its successors along the trusted path have received both the data d originating in O and E_n's endorsement based on previous endorsement dv_n, with security property sp. E_n is also obliged to prove all successive nodes have accepted delegation for non-repudiation. SPO uses a tree view as SSES allow data to be despatched to multiple intermediaries and recipients.

Sections 5.6.2.1 and 5.6.2.2 present assumptions and postulates used in destination and source proof obligations. Section 5.6.2.3 presents evidences needed to discharge destination proof obligations. Similarly Section 5.6.2.4 presents evidences needed to discharge source proof obligations. Section 5.6.2.5 presents attestable evidence for enforcing all two party security properties by extending proven two-party schemes presented in Section 3.6.3. Section 5.6.2.6 describes how evidences can be combined to discharge multiple proof obligations.

5.6.2.1 Assumptions

The assumptions presented in this section relate to trust server and compliant entities. These assumptions together with postulates presented in the next section form the basis for end-to-end schemes.

Trust Tree Assumptions (TTAs)

Trusted paths generated by the trust server in Chapter 4 reflect the category specific trust-related criteria specified. The following assumptions are made about trust trees.

TTA1: The trusted path forms part of the spanning tree from source to recipients.

TTA2: The trusted path consists of compliant entities and intermediaries only that meet the specified trust requirements.

TTA3: The centralized trust server (*TS*) has jurisdiction over trust relationships.

Valid Endorsement Assumption (EV)

Endorsement value dv_i in a compliant intermediary E_i must be set as $dv_i = f(E_i, DC, d, dv_{i-1})$ where d is data, dv_{i-1} is the previous-endorsement, DC is the data category and f is the predefined intermediary endorsement function reflecting legal requirements for endorsements in different domains and categories.

Proof Obligation Assumption (*POA*)

All compliant entities must discharge their proof obligations for the security properties required (specified by the trust server) or raise an exception. Proof obligations can be discharged directly or by transferring or delegating them. An exception may be raised when another entity fails to respond within stipulated time or because the data is stale or invalid.

Source Proof Obligation Assumption (*SPOA*)

Compliant intermediaries respond to a receipt of SPO from a valid predecessor and accept delegation to all nodes in the trusted path reached through them. Furthermore, an SPO can be delegated by a node only if it has received a valid delegation or if it is the source itself.

Destination Proof Obligation Assumption (*DPOA)*

A compliant entity in a trusted path (*Tree*) forwards data with a security property in {A,DI, TB} to its successor only if it has valid evidence to transfer or discharge its associated proof obligation.

5.6.2.2 Postulates

The postulates in this section extend the notation introduced for accountability logic in Section 2.5 of the background chapter. Postulates are presented in the form of a conclusion that can be deduced from a set of premises.

$$\frac{premise_1;\ premise_2\ ....\ ;premise_n}{Conclusion}$$

These postulates are used by interacting entities to reason and discharge any proof obligations.

Common Postulates

This section presents common postulates that form the basis for reasoning, similar to those presented in Section 2.5 for verifying accountability of existing protocols.

Statement: "*A says x*"

This construct makes A accountable for the statement x and anything implied by x.

Attestable Evidence from Digital Signature

Digitally signing a message provides attestable evidence of message origin. For example, when entity *X* receives a message *m* containing element *d*, signed with Y_{Pri}, *X* can prove that *Y* says *d*. Here *d* may refer to specific data, or specific semantics that have common predefined meaning. In the postulate below *X canAttest(p)* says that entity *X* can prove that the proposition *p* is true, based on the evidence presented.

$$\frac{X\ \text{Receives}\ (m\ SignedWith\ Ypri);\ d\ \text{in}\ m;}{X\ canAttest\ (Y\ says\ d)}$$

Attestable Evidence for Trust

If E_i receives a trust server signed digest containing a hash of data *d*, its data category *DC* and trusted path *Tree* rooted in *O* and having E_i as successor of E_{i-1}, E_i can deduce that the trust server says it can trust data *d* of data category *DC* originating in *O* and any endorsement (*dv*) related to it, from its predecessor E_{i-1}. It can be stated as in:

Trust:

$$\frac{Ei\ \text{Receives}\ (m\ SignedWithTSpri);\{d, Tree, DC\}\ \text{in}\ m;\ Next(Tree,Ei\text{-}1,Ei);\ Root(Tree,O)}{Ei\ canAttest\ (TS\ says\ \ trust(d + dv,O, MC,Ei-1,Ei\ ))}$$

5.6.2.3 Discharging Destination Proof Obligations

Destination proof obligations, which are initially placed on all data recipients in SSES, are discharged indirectly as all trusted paths consist of one or more intermediaries. Thus, proof obligations are transferred to the preceding intermediary, until it can be directly discharged with evidence from the data originator. Transferring proof obligations requires validating security property related evidence from the predecessor with evidence from the trust server. For example, in the Figure 5.6 below, the proof obligation for security property *sp* placed on *R* requires transferring it recursively to the predecessor specified by the trust server, until it can be discharged directly. This amounts to showing the scheme for security property *sp* is valid for both data *d* and endorsement dv_i as *i* varies from *n* down to *1*, and showing that the chain of endorsements dv_n, dv_{n-1}, … dv_1 are generated on a valid basis.. Each endorsement dv_i by intermediary E_i must be of the form $dv_i = f(E_i,DC,d,dv_{i-1})$ where *d* is the data and dv_{i-1} is the endorsement from the previous intermediary and *f* is the endorsement function. The first intermediary along the trusted path requires no endorsement; the term dv_0, a null endorsement, is introduced only for consistency of notation.

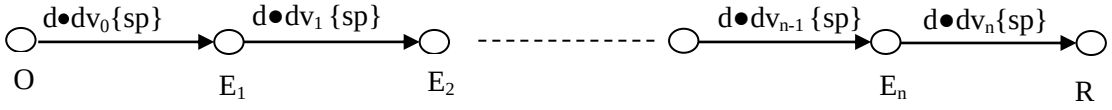


Figure 5.6 End-to-End Non-Repudiation through delegation

Attestable Evidence for Transferring or Discharging Destination Proof Obligations

In Figure 5.6, the destination proof obligation for *R*, $DPO[d+dv_n,O,DC,R,sp]$ can be transferred to the previous intermediary E_n in the server signed trusted path, subject to receipt of valid evidence from E_n for security property *sp*. Note the destination proof obligation for the previous intermediary is specified in terms of dv_{n-1} where $dv_n = f(E_i,DC,d,dv_{n-1})$.

$$\frac{R\ CanAttest\ (TS\ says\ (trust(d, O, DC,En,R));\ \ R\ CanAttest\ (En\ sent\ d \bullet dv_n\ with\ sp))}{(DPO\ [d \bullet dv_n,O,DC,R,sp]) \rightarrow (DPO\ [d \bullet\ dv_{n-1},O,DC,En,sp])}$$

In general, for each intermediary (other than the first one along the trusted path) destination proof obligation $DPO[d+dv_i,O,DC,E_i,sp]$ can be transferred to its predecessor along the trusted path, subject to evidence from the predecessor specified by the trust server.

If (i > 1)

TransferDPO:

$$\frac{E_i\ CanAttest\ (TS\ says\ (trust(d, O, DC,E_{i-1},E_i))\ ;\ E_i\ CanAttest(\ E_{i-1}\ sent\ d \bullet dv_{i-1}\ with\ sp\ )}{(DPO\ [d \bullet\ dv_i,O,E_i,sp]) \rightarrow (DPO\ [d \bullet\ dv_{i-1},O,E_{i-1},sp])}$$

For the intermediary immediately following the data originator, the proof obligation can be discharged based on direct evidence from the originator.

If (i = 1)

DischargeDPO: $\dfrac{E_1\ CanAttest\ (TS\ says\ (trust(d, O, DC, O, E_1)\ ;\ E_1\ CanAttest\ (\ O\ sent\ d \bullet dv_0\ )}{Discharge\ (DPO\ [d \bullet\ dv_1, O, E_1, sp])}$

The next section presents evidences needed for individual security properties.

5.6.2.4 Discharging Source Proof Obligations

Source Proof Obligations relate to security properties that provide assurances to the data source. End-to-end schemes for these properties can be derived by combining evidence needed to delegate them until they can be discharged directly. A specific security property, receiver non-repudiation (RNR) is used to illustrate SPO in this section as it is the only source security property considered in this chapter. In the example shown in Figure 5.7, non-repudiating *d* sent from *O* to recipients R_1 to R_5 requires involvement of intermediaries I_1 to I_4. Each node can discharge its obligation only after receiving evidence of data receipt and acceptance of delegation from all successors along the trusted path. Entity *O* requires I_1 to non-repudiate the receipt of data $d \bullet dv_O$ and acceptance of delegated obligation to non-repudiate $d \bullet dv_{I1}$ to all subsequent entities along that sub-tree (which in this case is only R_1) where new response dv_{I1} is formed by I_1 based on *d* and dv_0, i.e., $dv_{I1} = f(I_1, DC, d, d_{vO})$. Likewise, O requires I_2 to non-repudiate receipt of $d \bullet dv_O$ and accept delegation obligation to non-repudiate $d \bullet dv_{I2}$ to I_3 and I_4. This process of delegation is repeated until all recipients requiring non-repudiation are reached. These obligations are denoted by $SPO[d \bullet dv_{Ei}, O, DC, Ei, sp]$ (source proof obligation) where *Ei* is an entity that lies along the trusted path for data *d* of category *DC* originating in *O*, and dv_{Ei} is the endorsement from *Ei*.

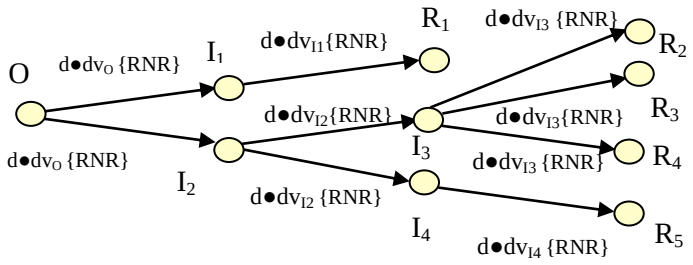


Figure 5.7 End-to-End Non-Repudiation through Delegation

The postulates for source proof obligation in this section relate to an entity *X* sending data $d \bullet dv_X$ to one or more successors along the trusted path, as shown in Figure 5.8. The sender *X* can either be the originator (in which case dv_X is empty) or an intermediary while the successors can either be recipients or intermediaries. In the Figure 5.8 *Next(Tree,X)* returns $N_1, N_2, \ldots N_i \ldots N_n$.

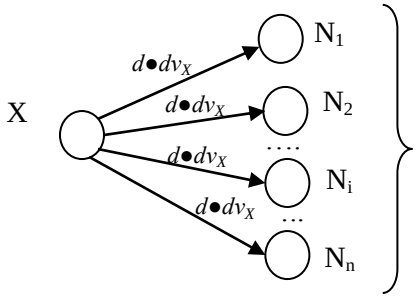


Figure 5.8 Successor Entities/Intermediaries along Trusted Path

Attestable Evidence for Delegating or Discharging Source Proof Obligations

Any intermediary or entity can delegate a non-repudiation proof obligation only if evidences for both data receipt and delegation acceptance are produced from all its successors in *Tree*. This results in non-repudiation proof obligation being delegated to all its successors. The transfer clause can be formally stated as shown below.

SPODelegate:

$$\frac{\forall\, N_i \in Next(Tree, X) : X\ CanAttest\ (N_i\ sent\ NR\_Evidence(d + dv_X) \wedge N_i\ AcceptDelegation(O, d+dv_{Ni}, SubTree(N_i))}{SPO\ [d + dv_X, O, DC, X, \{NR\}] \rightarrow \{SPO\ [d + dv_{Ni}, O, DC, N_i, \{NR\}]\ : \forall\, N_i \in Next(Tree, X)\ \}}$$

The non-repudiation obligation can be discharged directly when a leaf node of *Tree* is reached, as there are no further obligations.

SPODischarge:
$$\frac{Next(Tree, X) == \emptyset}{Discharge(SPO\ [d, O, MC, X, \{NR\}])}$$

5.6.2.5 Attestable Evidence for Enforcing Security Properties

This section presents attestable evidence for security properties that can be used to discharge proof obligations. These evidences are based on schemes presented as part of PGPS in Chapter 3. The recency property however, is replaced with the time-bound property which provides explicit evidence for timeliness. The non-repudiation property presented in Chapter 3 is extended to include delegation. All of these schemes allow protocols to be terminated halfway if data or sender cannot be validated or if property related assumptions cannot be met. The security properties enforced by all subsequent schemes are indicated indirectly using the index property index in Table 5.2.

Attestable Evidence for Authentication

Scheme

The scheme is identical to that presented in Section 3.6.3 with data consisting of data from the source and endorsement from the intermediary (sender). The proof is presented in Section 3.6.3.1.

Aborting Authentication

If the data or labels from the sender cannot be validated with the digest from the trust server *dig*, an exception message must be sent to the trust server to take corrective action, thus discharging any proof obligation.

Attestable Evidence for Data Integrity

Scheme

The scheme is identical to that presented in Section 3.6.3 with data consisting of data from the source and endorsement from the intermediary (sender). The proof is presented in Section 3.6.3.1.

Aborting Data Integrity

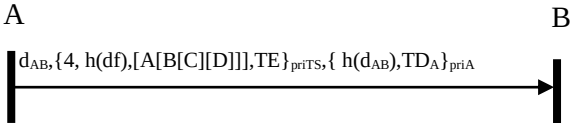
If the data or label from the sender cannot be validated with the digest from the trust server an exception message is sent to the trust server to take corrective action thus discharging any proof obligation.

Attestable Evidence for Time-Bound

Additional Assumption: Synchronization (SYN)

Time stamps between trusted intermediaries are generated using synchronized clocks.

Scheme



In the figure above, intermediary *B* can attest that data d_{AB} (formed combining data from the source with endorsement from *A*) was sent with the time-bound property from another entity *A* if the data hash $h(d_{AB})$ and the time of despatch TD_A was signed by its predecessor *A*, and the difference between despatch times (TD_B - TD_A) is lower than the expiry time (*TE*) singed by the trust server.

Time-Bound (TB) Scheme (*B*'s actions when receiving data from A with time-bound property)

Pre-condition: ***Honest(A,B, TS)*** **Invariants:** ***Inv1***	
L1. $Receive(B,A,[d_{AB},EMD_{TS},EMD_A])$	// receiving the data d and signed message digest EMD_3 *from A*
L2. $d \leftarrow d_{AB}.fixed$	// extract the data from source
L3. $MD_A \leftarrow Dec(B, EMD_A, pub_A)$	// *B* decrypts EMD_A with pub_A and stores it in MD_A
L4. $MD_A.hash=h(d)$	// verifying data hash
L5.$TD_A \leftarrow MD_A.TS$	// extract timestamp
L6. $MD_{TS} \leftarrow Dec(B, EMD_{TS}, pub_{TS})$	// *B* decrypts EMD_{TS} with pub_{TS} and stores it in MD_{TS}
L7. $MD_{TS}.hash=h(d)$	// verify data hash
L8. $TE \leftarrow MD_{TS}.TE$	// extract maximum time allowed
L9. $TD_B \leftarrow Timestamp()$	// *B* time-stamps current time
L10. $TD_B - TD_A \leq TE$	// verify the difference between despatch times is lower than *TE*
Pre-condition: ***Honest(A,B, TS)*** **B Assert timeliness of data from A**	
Invariants: ***Inv1***	

Proof: Time-bound Scheme allows B to verify Timeliness of A's Data

By precondition ***Honest(***A,B,TS***)*** and honest entity assumption **HEA** (Table 3.12), invariant **Inv1** is initially true. By axiom PA7 (Table 3.11), MD_A in **L3** derived decrypting message EMD_A by public key of A, was despatched by an entity in possession of private key of A. Based on the precondition, A is honest and by assumption **HEA Inv1** holds (private keys are kept confidential), it follows MD_A originated in A. **L4** and **L5** together with the precondition ***Honest(A)*** allows time of data (d) despatch from A to be extracted. By axiom PA7, MD_A in **L6** derived decrypting message EMD_{TS} by public key of TS, was despatched by an entity in possession of the private key of TS. Based on the precondition TS is honest and by assumption **HEA** (Table 3.12) **Inv1** holds (private keys are kept confidential), it follows MD_{TS} originated in TS. **L7** and **L8** together with the precondition ***Honest(***TS***)*** allows expiry time TE from the trust server to be extracted. **L9** and **L10** together with the additional assumption clocks are synchronized allows timeliness of data from A to be asserted**.** □

Aborting Time-Bound

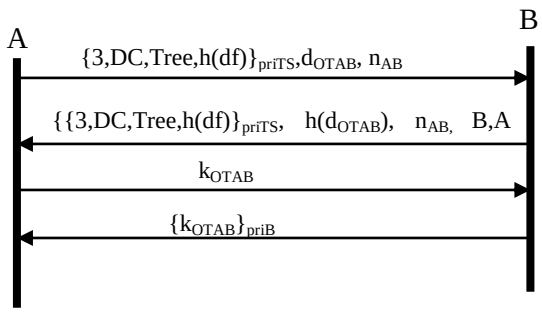
If the data from the sender cannot be validated with the digest from the trust server, or if the sum of node and communication delay exceeds TE an exception message is sent to the trust server to take corrective action, thus discharging any proof obligation.

Attestable Evidence for Non-repudiation and Delegation

Additional Assumption: Delegation Assumption (DA)

The recipient sending an acknowledgement of server signed digest containing a hash of the data with the hash of the tree representing trusted path amounts to accepting delegation for all subsequent nodes along the tree.

Extended Scheme for Non-Repudiation (including Delegation)



The data d sent from A combines data from source d_f with the endorsement by A, dv_A. The scheme differs from the non-repudiation scheme used in Chapter 3 in terms of sending the server signed digest ($\{3,DC,Tree,h(df)\}_{priTS}$) in step 1 and receiving it as part of the signed term in step 2. This additional step is used for getting acceptance of delegation for all subsequent nodes.

Proof: Extended Scheme allows Delegation Acceptance in addition to Data Non-Repudiation.

This is an extension of the RNR Proof Presented in Section 3.6.3.1. Only differences are in L4 and L7.

Pre-condition: ***Honest(A,B)*** **Invariants:** ***Inv1,Inv2,Inv3,Inv4***

…

L4. *Send(A,B,[dOTA,AnB,**SD**)* // *A* sends encrypted data, nonce and the server signed digest *SD*

L5. *Receive(A,B, EMD_4)* // receive encrypted digest containing data hash, nonce and labels

L6. $MD_4 \leftarrow Dec(A, EMD_4, pub_B)$ // *A* decrypts encrypted message digest storing it in MD_4

L7. $MD_4.hash=h(d_{OTA})$ $MD_4.nonce={}_An_B$ $MD_4.sender=B$ $MD_4.receiver=A$ $MD_4.SD = \mathbf{SD}$

// contains the additional term – SD sent in L4

Post-condition: ***Honest(A,B); A Assert(Message Non-Repudiation and Acceptance of Delegation by B)***

Invariants: ***Inv1,Inv2,Inv3,Inv4***

Proof: Non-repudiation of Delegation (in addition to Data Receipt proved in 3.6.3.1)

Presence of trust server signed digest containing the trusted path in the recipient signed term in **L7** amounts to accepting delegation to all subsequent nodes along the tree based on the assumption DA.

Aborting Non-repudiation

If one of the child nodes does not respond within a predefined timeout period, an exception message is raised to the trust server, passing on the sender, non-responding entity, security property and cause. The timeout period is predefined though time-bound property can be combined with non-repudiation to specify appropriate value for a particular application context.

Enforcing Secrecy

The accountability logic does not allow reasoning about techniques used for enforcing secrecy [30]. Therefore a scheme independent of others using symmetric group keys is devised to enforce end-to-end secrecy in SSES. These schemes are enforced at a fine-grained knowledge-element level to allow access to different parts of data to be restricted to specific entities. Two other mechanisms for enforcing secrecy include multiple key ciphers [101] and the public key systems [85]. The multiple key ciphers system is appropriate when the number of entities sharing secret s is large, as the number of keys required is the same as the number of entities. Public key systems are computationally more expensive than symmetric keys. Protocol bandwidth, too, may increase substantially if each secret knowledge element is encrypted by all the public keys of intended recipients. For example, if *A* is sending data to *B, C* and *D* through trusted intermediaries *E* and *F*, three distinct messages must be sent through *E* and *F*, each encrypted with public keys of *B, C* or *D*. Bandwidth can be reduced if a common public key scheme is used. A symmetric group key scheme however, requires sending only one copy of secret data encrypted by the common group key, though it requires a symmetric group key server such as ELK to be present for distributing symmetric group keys [103]. An entity requesting a group key sends a message containing the list of entities in the group signed by its private key. The key server creates a new one if one does not already exist for the specified group or if it has expired.

5.6.2.6 Discharging Multiple Proof Obligations

Proof obligations for multiple security properties can be discharged by combining attestable evidences that are non-interfering. For example, in Figure 5.4 evidences needed for discharging security properties *{A}, {TB}* and *{RNR}* along the data path were combined. The protocol composition rule states that two schemes can be combined in parallel if the invariants for one does not interfere with the invariants for the other [25]. For example, the validity of a scheme relying on the invariant "a private key cannot be divulged" cannot be combined safely with another scheme sending the same key in the open. More formally, two schemes Q and Q' with invariants Γ and Γ', can be combined in parallel if they do not violate the invariants $\Gamma \cup \Gamma'$ (refer to Section 2.3.2.2). In Chapter 3, non-interfering cryptographic schemes for properties were combined directly to provide multiple security assurances. In a similar way, SSES combines non-interfering schemes that provide explicit evidence.

When invariants for basic schemes interfere, non-interfering schemes must be devised before they can be combined, as in Chapter 3. For example, when combining with non-repudiation scheme, extended schemes for authentication, time-bound and data-integrity are used instead, as the standard schemes violate the invariant Inv3 assumed to hold by the non-repudiation scheme. For authentication and data integrity, the basic and extended schemes are identical to those used in Chapter 2 as they also provide explicit evidence. For the time-bound property which replaces recency, a revised version is created by first releasing data encrypted by a one-time-key followed by the one-time-key on receipt of an acknowledgement.

5.6.3 Role of Proof Obligations

The trust server determines the trusted paths and the path security levels aggregating the trust and the security requirements specified. Each node along the trusted path is required to discharge its proof obligations for the security properties assigned by obtaining evidence from its neighbouring nodes. If all interacting entities and intermediaries can discharge their proof obligations end-to-end trust and security can be guaranteed. If however, such obligations cannot be discharged the protocol run may be aborted midway after alerting the trust server to from a different trusted path. Even if all proof obligations are discharged an entity or intermediary can be held liable if the underlying assumptions are found to have been violated. A process similar to audit can be carried out with proof obligations to identify the noncompliant node. Any entity or intermediary failing to produce the required (archived) evidence for the underlying assumptions can be held liable. It is assumed archived data is stored for a predefined period of time. These three possible scenarios are outlined briefly in this section.

Detecting Violations during Protocol Execution

Validity of data can be verified directly by comparing it with the hash of data in the digest sent from the trust server. If data cannot be validated the protocol may be aborted. When enforcing specific properties such as time-bound or receiver non-repudiation, long communication delays may cause protocol execution to be terminated.

End-to-End Security and Trust

Using trust tree assumptions *TTA1*, *TTA2* and *TTA3* the trusted path consists of compliant entities that meet the trust requirements. If no exceptions are raised, then based on assumptions *POA*, *SPOA* and *DPOA*, all compliant entities or intermediaries have discharged their source and destination proof obligations, meaning data and its successive endorsements have reached their intended recipients along the trusted path. Thus, when all proof obligations are discharged without any exceptions, end-to-end trust and security requirements are met for all data and endorsements.

Detecting Non-Compliant Entities

Every entity and intermediary involved in discharging a proof obligation directly or indirectly is required to adhere to the underlying assumptions for compliant entities and intermediaries. Any entity or intermediary failing to produce the necessary (archived) evidence is considered to have breached trust assumptions, which may lead to severance of all trust relationships. For example, an invalid endorsement that does not comply with $dv_n = f(E_i,DC,d,dv_{n-1})$ is considered to have violated the assumption *EV*. Similarly an intermediary enforcing time-bound property is considered to have violated a trust assumption if the difference between its despatch time and its predecessor despatch time exceeds the server specified time limit.

5.6.4 Example: End-to-End Schemes Derived from Proof Obligations

This section presents examples of end-to-end schemes derived by combining two-party evidence used for discharging, delegating or transferring proof obligations along a trusted path involving two recipients and one intermediary. Both *DPO* and *SPO* also specify preconditions for all intermediaries forwarding messages. These schemes implicitly show the preconditions which are based on the assumptions. For example, an intermediary cannot send authenticated data until and unless it has received the data authenticated (Assumption *DPOA*). Similarly an intermediary cannot delegate the non-repudiation property until that property has been delegated to it (Assumption *SPOA*). Schemes for basic properties can be combined to provide multiple security properties, as long as the individual schemes for basic properties do not violate each other's invariants [61]. Example schemes for multiple properties are presented in the next section.

Symbols used in the Schemes

The schemes presented in this section and the next use the trusted path presented in Section 5.5.4 where:

- trusted path from *A* to data recipients *C* and *D* through intermediary *B* is *[A[B[C][D]]]*. The term *Tree* is the abbreviation for [*A*[*B*[*C*][*D*]]] in Figures 5.9 to 5.16, indicating that data from *A* must be sent to entities *C* and *D* via *B*.
- the symbol *df* represents data from its origin *A*
- d_{BC} represents the combination of data *df* and endorsement dv_{BC}
- *SPI* (1..15) represents the security properties index (refer to Table 5.2)
- *DC* is the data category
- *TS* is the trust server
- $\{SPI,DC,h(df),[A[B[C][D]]]\}_{priTS}$ is the trust server signed digest

5.6.4.1 End-To-End Authentication Scheme

The end-to-end authentication scheme derived from discharging the destination proof obligation along trusted path [*A*[*B*[*C*][*D*]]] is shown in Figure 5.9. Each of the two-party schemes making up the end-to-end scheme consists of two digests. The first digest consists of the IDs for sender and destination as well as the combined hash for fixed and variable parts, while the trust server signed digest consists of *SPI*, *DC*, a hash of fixed data and the trusted path. Note that the protocol path *A*→*B*, *B*→*C* and *B*→*D* in Figure 5.9 lie along the trusted path [*A*[*B*[*C*][*D*]]] specified by the trust server. In addition, the data combining fixed and variable parts in plaintext is shown as one term ($d_{BC} = df + dv_{BC}$). Intuitively, recipient *C* can verify the validity of the sender based on the server signed trusted path, and verify the association between two digests based on the common hash of fixed data contained therein.

Assumptions: TTA1, TTA2, TTA3, EV, POA, SPOA

Invariants: Inv1

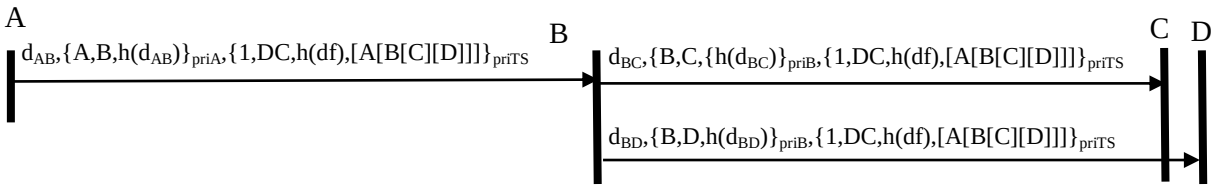


Figure 5.9 End-to-End Scheme for Authentication

5.6.4.2 End-to-End Data Integrity Scheme

The derived scheme for data integrity shown in Figure 5.10 is identical to that of authentication, except for the omission of a label specifying data destination.

Assumptions: TTA1, TTA2, TTA3, EV, POA, SPOA

Invariants: Inv1

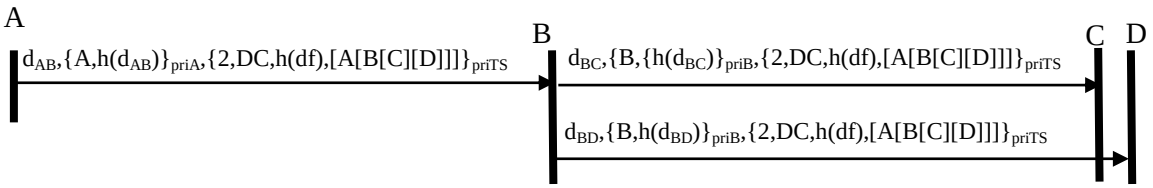


Figure 5.10 End-to-End Scheme for Data Integrity

5.6.4.3 End-to-End Time-Bound Scheme

The scheme derived for time-bound property, shown in Figure 5.11, contains two signed digests for each step: one for data and one for endorsement. Note that each digest signed by the preceding intermediary or entity consists of the data despatch time (such as TD_A,TD_{B1},TD_{B2}), while the one signed by the trust server consists of the data expiry time (*TE).* An intermediary must not forward data that has already expired, to avoid despatch of stale data. However, the intermediary is expected to raise an exception to the trust server allowing corrective action. The time-bound property prevents replay attacks by limiting the time window in which the messages are considered valid.

Assumptions: TTA1, TTA2, TTA3, EV, POA, SPOA, SYN

Invariants: Inv1

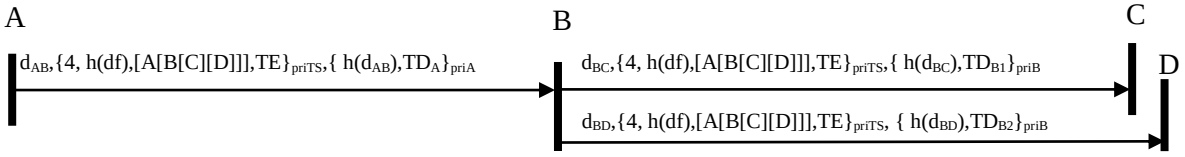


Figure 5.11 End-to-End Scheme for Time-Bound Property

5.6.4.4 End-to-End Receiver Non-Repudiation Scheme

The scheme derived for the non-repudiation property allows partial release of data as shown in Figure 5.12. In the initial pass, a server signed message consisting of the trusted path, a hash of the fixed part, the data encrypted with a one-time key (such as d_{OTAB}) and the nonce n_d (sent from message-origin to distinguish one run from another) are sent. The term d_{OTAB} is formed by encrypting d with a one-time key k_{OTAB}, where d is the string formed appending the variable and fixed segments ($d_{OTAB} = (df \bullet dv_{AB})_{KOTAB}$). The key k_{OTA} is despatched only after signed acknowledgement from the recipient, made up of the signed trust tree, a hash of the encrypted message and a nonce. If successor acknowledgement is not received within a pre-set time limit, an exception is sent to the trust server, which acknowledges receipt, thus discharging any liability. This scheme ensures that recipients cannot deny data receipt, as the sender can show the acknowledgement to an arbitrator as evidence of receipt. Furthermore, recipients are required to forward data to all subsequent nodes along the trusted path.

Assumptions and Invariants: A1, A2, A3, Inv1, Inv2, Inv3

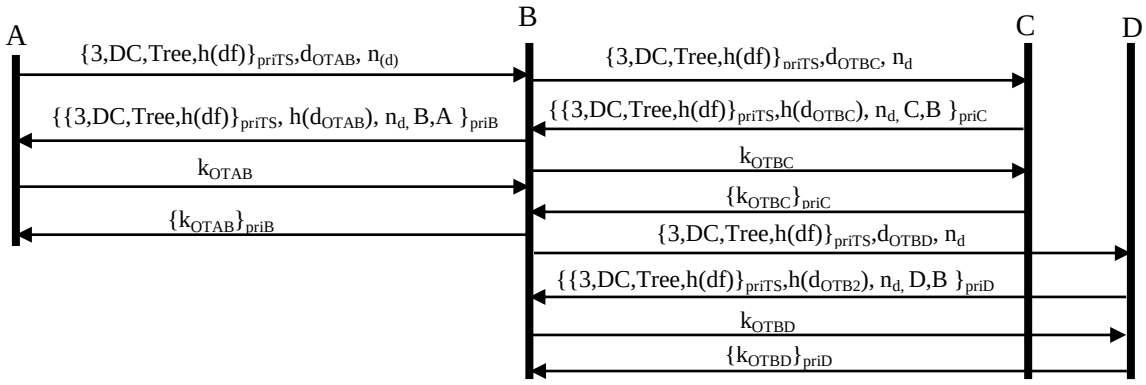


Figure 5.12 End-to-End Scheme for Non-Repudiation

5.6.5 Example: Composed End-to-End Schemes

This section presents the composed schemes using the methodology outlined in Section 5.6.2.5 which requires extended schemes to be combined when basic schemes interfere.

5.6.5.1 End-to-End Scheme Combining Authentication and Time-Bound

Figure 5.13 shows the scheme meeting authentication and time-bound properties. This scheme, though costlier than the authentication scheme alone, can be used when replay attacks needs to be prevented. In this scheme each entity or intermediary receives the time of despatch from the previous intermediary along the trusted path, signed by its private key, and the data expiry time signed by the trust server, in addition to all the other elements included in the authentication scheme. These elements, together with non-conflicting invariants for authentication and the time-bound property ensure that the combined scheme meets both authentication and time-bound goals (using the composition rule).

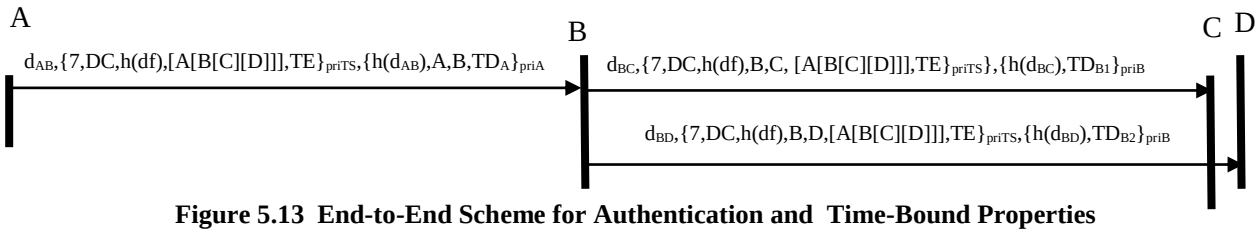


Figure 5.13 End-to-End Scheme for Authentication and Time-Bound Properties

5.6.5.2 End-to-End Scheme Combining Authentication and Non-Repudiation

The combined scheme shown in Figure 5.14 was arrived at by combining the non-repudiation scheme with an extended (non-conflicting) authentication scheme. End-to-end protocols combining authentication and non-repudiation allow data recipients to verify that data was sent through authenticated channels before providing non-repudiation evidence. This results in an additional signed term in the first step of the non-repudiation scheme associate data with message origin and intended destination. Note that the last term in the first step from *A* to *B, B to C and B to D* in Figure 5.14, is a sender signed term made up of a hash of encrypted data, source and destination information, as well as the nonce used in non-repudiation.

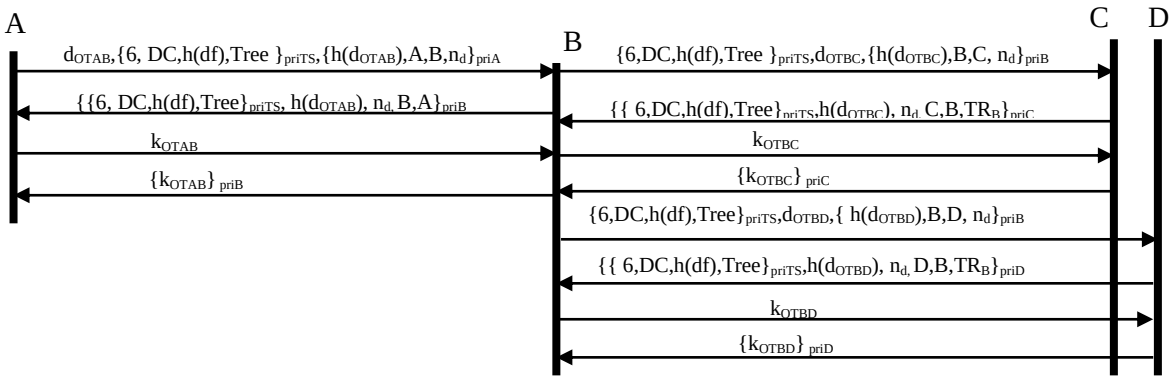


Figure 5.14 End-to-End Scheme for Authentication and Non-Repudiation

5.6.5.3 End-to-End Scheme Combining Non-Repudiation and Time-Bound

The scheme combining the non-repudiation and extended time-bound schemes is shown in Figure 5.15. This combined scheme allows recipients to verify that the message is timely before acknowledging it. The message originator gets non-repudiable evidence of timely message receipt. Such a composite property, for example, may be used when bidding for a business tender that has a specific expiry date/time.

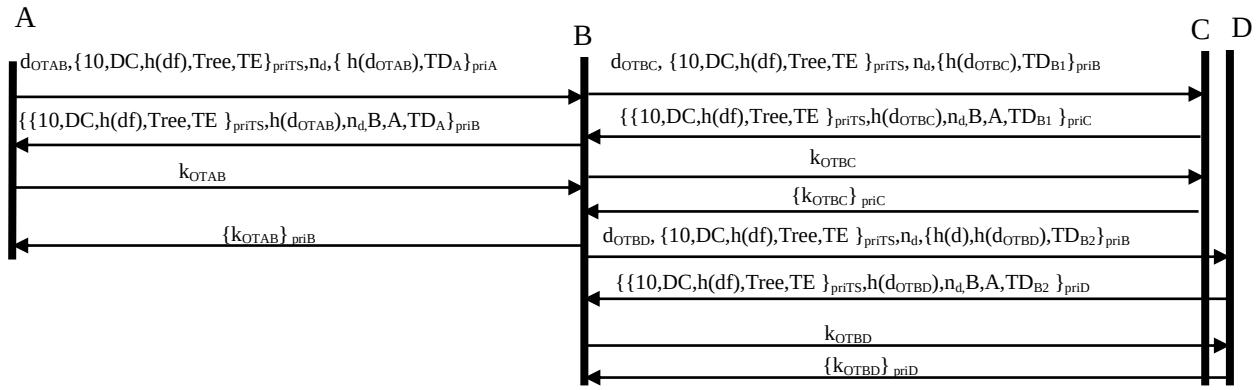


Figure 5.15 End-to-End Scheme for Non-Repudiation and Time-Bound Properties

5.6.5.4 End-To-End Scheme Combining Authentication and Time-Bound

The extended time-bound and authentication schemes can be combined with the non-repudiation scheme to provide a stronger property, as shown in Figure 5.16.

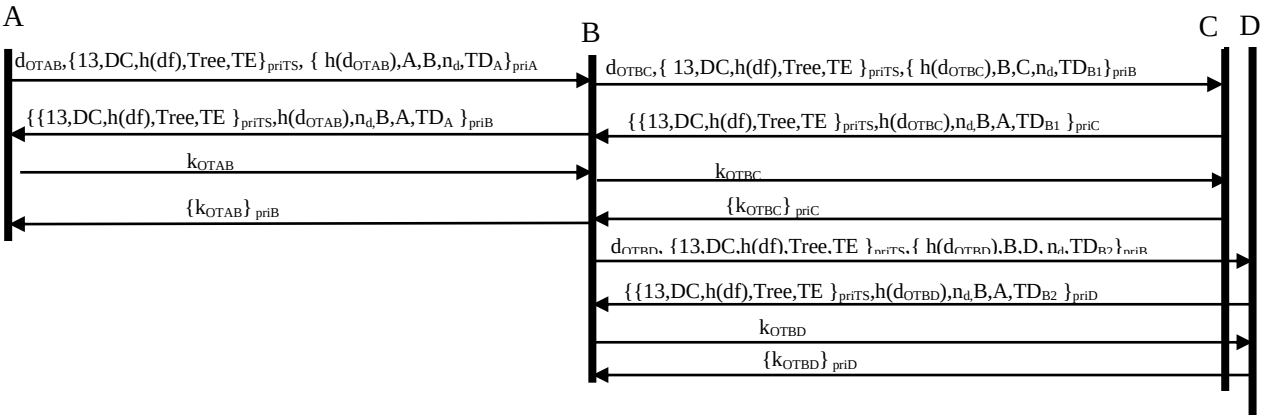


Figure 5.16 End-to-End Scheme for Authentication, Non-Repudiation and Time-Bound properties

5.6.5.5 Summary of Fine-grained End-to-End Security Schemes

All the security properties, including the combined ones, are summarised in Table 5.3 based on data *d*, origin *O*, sender *S*, receiver *R*, security-level *sl* and data trust tree *Tree*. Note there can be up to four hops for a given security-level. The abbreviation *TreeN* is used for $\{N,DC,h(df),Tree, \ldots\}_{priTS}$ where *N* is the index for the security property. For example, when *N=1*, *TreeN* is $\{1,DC,h(d),Tree\ \}_{priTS}$ and when *N=13*, *TreeN* is $\{13,DC, h(d),Tree, TE\}_{priTS}$ where TE is the time of data expiry. Note that only those properties that include the time-bound property *(N=4,7,9,10,12,13,14,15)* have the additional element *TE*.

N	Security Properties	Security Scheme
1	{A}	$S[d_{SR}, TreeN, \{S, R, h(d_{SR})\}_{priS}]R$
2	{DI}	$S[d_{SR}, TreeN, \{S, h(d_{SR})\}_{priS}]R$
3	{RNR}	$S[TreeN, d_{OTSR}, \{h(d), h(d_{OTSR}), n_d\}_{priS}]R \bullet R[TreeN, \{h(d_{OTSR}), n_{d,} R, S\}_{priR}]S \bullet S[k_{OTSR}]R \bullet R[\{k_{OTSR}\}_{priR}]S$
4	{TB}	$S[d_{SR}, TreeN, \{h(d_{SR}), TD_S\}]R$
5	{A,DI}	$S[d_{SR}, TreeN, \{S, R, h(d_{SR})\}_{priS}]R$
6	{A,RNR}	$S[TreeN, d_{OTSR}, \{h(d_{OTSR}), S, R, n_d\}_{priS}]R \bullet R[\{TreeN, h(d_{OTSR}), n_{d,} R, S\}_{priR}]S \bullet S[k_{OTSR}]R \bullet R[\{k_{OTSR}\}_{priR}]S$
7	{A,TB}	$S[d_{SR}, TreeN, \{h(d_{SR}), S, R, TD_S\}_{priS}]R$
8	{DI,RNR}	$S[d, TreeN, d_{OTSR}, \{S, h(d_{OTSR})\}_{priS}, n_d]R \bullet R[\{TreeN, h(d_{OTSR}), n_{d,} R, S\}_{priB}]S \bullet R[k_{OTSR}]S \bullet R[\{k_{OTSR}\}_{priR}]S$
9	{DI,TB}	$S[d_{SR}, TreeN, \{\{h(d_{SR}), TD_S\}_{priTS}\}R$
10	{RNR,TB}	$S[d_{OTSR}, n_d, TD_A, TreeN \{h(d_{OTSR}), TD_S\}_{priS}]R \bullet S[\{TreeN, h(d_{OTSR}) n_{d,} R, S, TD_S\}_{priR}]R \bullet S[k_{OTSR}]R \bullet R[\{k_{OTSR}\}_{priR}]S$
11	{A,DI,RNR}	$S[d_{OTSR}, TreeN, \{h(d_{OTSR}), S, R, n_d\}_{priS}]R \bullet R[\{TreeN, h(d_{OTSR}), n_{d,} R, S\}_{priR}]S \bullet S[k_{OTSR}]R \bullet R[\{k_{OTSR}\}_{priR}]S$
12	{A,DI,TB}	$S[d_{SR}, TreeN, \{h(d_{SR}), S, R, TD_S\}_{priTS}]R$
13	{A,RNR,TB}	$S[d_{OTSR}, TreeN, \{h(d_{TSR}), S, R, TD_A, n_d\}_{priS}]R \bullet R[\{TreeN, h(d_{OTSR}), n_{d,} R, S, TD_S\}_{priR}]S \bullet S[k_{OTSR}]R \bullet R[\{k_{OTSR}\}_{priR}]S$
14	{DI,RNR,TB}	$S[d_{OTSR}, TreeN, \{h(d_{TSR}), TD_A, n_d\}_{priS}]R \bullet R[\{TreeN, h(d_{OTSR}), n_{d,} R, S, TD_S\}_{priR}]S \bullet S[k_{OTSR}]R \bullet R[\{k_{OTSR}\}_{priR}]S$
15	{A,DI,RNR,TB}	$S[d_{OTSR}, TreeN, \{h(d_{OTSR}), S, R, TD_A, n_d\}_{priS}]R \bullet R[\{TreeN, h(d_{OTSR}), n_{d,} R, S, TD_S\}_{priR}]S \bullet S[k_{OTSR}]R \bullet R[\{k_{OTSR}\}_{priR}]S$

Table 5.3 End-to-End Schemes for Authentication, Non-Repudiation and Time-Bound Properties

5.6.6 Reducing the Cost of End-To-End Security Schemes

This section describes two techniques employed by SSES to reduce the overall computational and bandwidth costs of messages sent to multiple recipients. First, it shares common intermediaries and endorsements, as shown in Figures 5.9 to 5.16. Second, the security level along any path from data originator to recipient cannot be raised. The actual security level is computed as a function of security requirements at data originator and recipients, as described in Section 5.6.6.1. This technique, however, requires each entity and intermediary to store the necessary schemes for enforcing every combination of security properties. Section 5.6.6.2 present the technique devised for reducing the size of string (metadata), which contains the trusted path and the required security levels.

5.6.6.1 Variable Security Level for Reduced Overheads

The security level along any edge is computed by aggregating the security requirements of recipients reached through that edge and the security requirement specified by the data originator. In SSES the security level cannot be raised along any path as additional evidences may be required from the data originator. For example, assume data originating in *A* must be sent to *C* and *D* through intermediary *B*, and that the security requirements at *A, B* and *C* are as follows: *A:{RNR}, C:{R}, D:{A}*. Then the minimum security levels along *AB*, *BC* and *BD* are respectively *{A,RNR,R}, {RNR,R}* and *{RNR,A}*. Assigning the minimum security level along data path helps to lower the cryptographic overheads. Then next algorithm automatically assigns the minimum security level without affecting end-to-end security.

Overview and Description of Security Level Assignment

SLA(*tpath, oSP, resSPs*,) computes the security level for each edge along the trusted path *tpath*, where *oSP* specifies the security level for the message-originator, and *resSPS* the security levels for recipients. For example, an argument to SLA could consist of the string "[*A*[*B*[*C*][*D*]][*E*[*F*][*G*]]]" for the trusted path *tpath*, *{DI}* for *oSP*, and [(*C*,*{RNR}*), (*F*,{*A*,*RNR*})] for *resSPs*. The algorithm first constructs the spanning tree from the trusted path (string) specified before assigning the security property *oSP* to all edges. It then adds to each edge along the tree the security properties of all recipients in *resSPs* reached through that edge. Finally, it forms *stpath* (string) the trusted path with security levels encoded.

Input: *tpath, oSP, resSPs*

Output: *stpath*

==========

1. form the *spanning tree* based on input (string) *tpath.*
2. add the security property *oSP* along all edges of the tree *PTree*.
3. for each recipient in *resSPs*

   for each edge in *PTree* from origin to the specified recipient

   add the associated security properties.
4. form the security encoded trusted path from the spanning tree.

==========

Table 5.4 Security-Level Assignment Algorithm

The result of applying this algorithm for the specified input is shown in Figure 5.17.

Input

tPath (String)
"[*A*[*B*[*C*][*D*]][*E*[*F*][*G*]]]"

oSP (Security Property for Originator)	
Originator	Security Properties
A	{DI}

resSPs (Security Properties for Recipients	
Recipient	Security Properties
C	{RNR}
D	{}
F	{A,RNR}
G	{}

Output

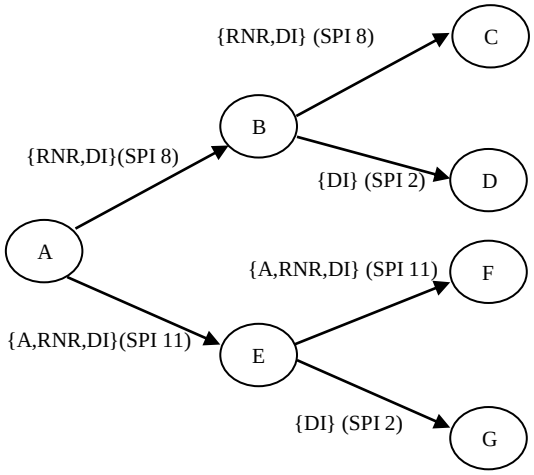


Figure 5.17 Minimum Security-Level Based on Fine-Grained Security Requirements

Providing End-to-End Assurances

SSES allows compatible two-party schemes along the path to be combined to meet end-to-end security assurances. In Figure 5.17 the *{A,RNR,DI}* scheme from *A* to *E* can be combined with the *{DI}* scheme from *E* to *G* to provide end-to-end security assurance *{DI}* between *A* and *G*. In general if for each data path from originator to recipient the security level enforced exceeds p_l, the end-to-end assurance p_l can be provided for that data. This is because each intermediary along the path uses a scheme that contains the necessary evidence to discharge its proof obligation. Composite schemes used for providing multiple properties do not interfere with schemes for basic properties. Thus, composite schemes for multiple properties can be combined to provide end-to-end security assurance for common properties.

5.6.6.2 Space Efficient Encoding of Variable Security Levels in Trusted Path

In SSES, variable security levels are encoded along edges of the trusted path (string) by the trust server. The size of such strings is reduced by setting the security level to an edge only if it is different from its previous edge. For example, the tree structure in Figure 5.17 is encoded into security encoded trusted path *stpath* as in "[*A*[*B:8*[*C*][*D:2*]][*E:11*[*F*][*G:2*]]]". If the security level is not explicitly specified when decoding the trusted path (to determine the required security level), it is set to the closest edge up the tree with an explicit security level. In "[*A*[*B:8*[*C*][*D:2*]][*E:11*[*F*][*G:2*]]]", security level along *EG* is *{DI}* based on 2 (explicitly specified), and *EF* is *{A,RNR,DI}* (implicitly specified) based on the *SPI* 11 assigned to the next closest edge (*AE*) up the tree.

5.7 Discussion

The framework presented in Chapter 4 helped to alleviate the perception of risk in e-commerce by endorsing key messages. However, such endorsements introduce additional security and performance challenges. Some of these problems such as the need for accountability, security for data originating in a source and intermediaries and reducing the high security overheads are common to many other domains. The following sections discuss the techniques devised to address some of these challenges.

5.7.1 Security Schemes Incorporating Intermediary Endorsements

End-to-end schemes in SSES provide fine-grained security guarantees for data originating in source and endorsements made by intermediaries along the trusted path. Valid endorsements are computed on the basis of data from the source, data category and previous endorsements if any. Path security is enforced by validating data from intermediaries and entities with server signed digest containing the trusted path and the hash of data. The proposed solution has a number of benefits. In the past high overheads had made it difficult to secure data sent through content transformation intermediaries [146]. SSES reduces security overheads by using fine-grained schemes that provide the right level of security. Furthermore, SSES allows minimum security levels to be used along each edge of the trusted path based on security requirements of data recipients using that edge. SSES also reduces endorsement costs by allowing data for multiple recipients to be sent through common endorsers. Schemes for content transformation intermediaries must also prevent unauthorised access and alterations [145]. SSES server signed digest ensures data from source cannot be tampered by intermediaries while SSES secrecy schemes allow additional data elements to be sent with restricted access. The main drawback of this technique, however, is the reliance on a common trust server for all data needing endorsement. Furthermore assumptions about synchronized clocks between trusted nodes may be difficult to enforce in practice unless logical clocks and timestamps are used.

5.7.2 Synthesising End-to-End Schemes from Proof Obligations

Maintaining centralized trust networks (used in Chapter 4) require end-to-end accountability schemes that can detect any misconduct. Accountability schemes provide the explicit evidence necessary to discharge any liability in e-commerce [155]. The end-to-end accountability schemes in this chapter were derived based on the notion of proof obligations for security properties. Proof obligations allow entities and intermediaries to reason about explicit evidences needed to enforce accountability. Any intermediary failing to produce the necessary evidence to transfer, delegate or discharge its proof obligation is considered to have breached trust assumptions. Destination proof obligations are placed on all recipients while source proof obligations are placed on data originators. The work done in this thesis deriving accountability schemes differs from past work done, which was primarily used to verify whether existing protocols enforce accountability [30, 155]. The applicability of this new

approach is not limited to endorsement intermediaries alone; accountability schemes are essential for web services, cloud based solutions and content transformation intermediaries.

5.7.3 Mechanism for Enforcing End-to-End Schemes

If e-commerce entities are to interact dynamically, the end-to-end schemes derived must be made available to all interacting entities and intermediaries. The solution presented in SSES uses trust server signed metadata containing trusted path and path security levels as a form of protocol description. The trust server determines the trusted path on the basis of existing trust relationships and the trust requirements specified by the data originator and recipients. The data originator despatches the trust server signed metadata containing the protocol description together with the data. Each entity or intermediary along the trusted path enforces the specified security level using predefined two-party schemes which are combined to meet the end-to-end security requirements at runtime.

5.7.4 Future Work

This chapter does address problems related to storage, retrieval and revocation of evidences. Also the current model reliance on a single trust server can lead to performance bottlenecks and potential loss of all data in cases of any security compromise. Hence, a distributed model for trust services must be explored. Endorsement functions too are likely to change with time making it necessary to maintain different versions to perform the necessary audit.

5.8 Conclusion

End-to-end security is a major research challenge facing content transformation systems, web services and e-commerce. Many past solutions for end-to-end security appear to be inadequate partly because they do not explicitly capture the interdependencies between trust and security. The proposed framework combining trust and security aspects has two main benefits. First, endorsement intermediaries can be chosen on the basis of dynamically evolving trust relationships. Second, trust and endorsement breaches can be detected using end-to-end accountability schemes. Although a number of standard protocols incorporate accountability, no past technique allows end-to-end accountability schemes to be derived for a given configuration. In particular, the following contributions were made.

- End-to-end schemes devised provide security for both data from source and endorsements by intermediaries. The server signed metadata consisting of the trusted path and data hash allows intermediaries to validate the data and endorsement path before performing their own endorsements.

- End-to-end schemes were derived directly from proof obligations. End-to-end assurances follow directly when all proof obligations are discharged. Intermediaries are made accountable as the cause of transaction failure can be traced to a trust breach or invalid endorsement of a specific intermediary. Any invalid endorsement or trust breach can be proved to a third party using explicit cryptographic evidences included in the end-to-end schemes.

- A novel mechanism allowed end-to-end security schemes to be enforced dynamically by combining the metadata consisting of trusted path and security levels with sequentially composable two-party schemes. Security overheads were reduced by using minimum security levels along the path, which are derived by aggregating the security requirements of data originator and recipients.

Chapter 6: Conclusion and Future Work

This thesis has considered the security problems facing e-commerce. Security protocols must be synthesised on-the-fly if e-commerce entities with specific security requirements are to collaborate dynamically. Before security protocols can be synthesised, however, security properties themselves must be defined unambiguously. If security protocols are to be synthesised dynamically for e-commerce the basic properties must be made to incorporate features such as fair exchange. Furthermore, cryptographic protocols for e-commerce must be designed to work within resource constraints and bandwidth limitations.

Trust has always played a major role in security protocols. Kerberos and other protocols using public key systems rely on trust placed on servers and certification authorities respectively. Such a binary trust model, however, is inadequate for modelling trust relationships that exist in the real world. E-commerce currently lacks an institutional framework where trust between unknown entities can be established through category specific endorsement intermediaries. Use of such intermediaries is predicated on devising end-to-end security mechanisms that make all intermediaries accountable for their actions. Such end-to-end security schemes are also vital for web services and content transforming intermediaries where messages are passed through one or more intermediaries. The solutions presented allowed:

- aggregation of end-to-end security requirements using operations defined over fine-grained security properties
- creation of security protocols at runtime combining provably correct security schemes devised for fine-grained security properties
- schemes for delivering data to any number of recipients by interleaving two-party schemes
- techniques to make security performance trade-offs by estimating computational cost and bandwidth based on underlying cryptographic strength
- reducing the risk of trading with unknown e-commerce entities using category specific endorsement intermediaries
- end-to-end schemes that make intermediaries along the trusted path accountable for their actions
- end-to-end schemes that provide guarantees for both data and endorsements along the path
- self-regulating trust relationships for endorsement intermediaries based on accountability schemes and trust evolution/transfer policies
- a mechanism to enforce end-to-end schemes dynamically

6.1 Research Contributions

The solutions presented include fine-grained security properties and schemes, a cost model that allows trade-offs between security and cost, an institutional trust framework for building trust between unknown e-commerce entities, end-to-end schemes that make all intermediaries accountable and a mechanism for enforcing end-to-end schemes dynamically. The specific contributions are presented below.

A New Mechanism for Quantifying and Reasoning About Security Requirements

The fine-grained security properties proposed organize security requirements into hierarchical security levels using a lattice model. These security levels were formed by combining basic security properties providing assurances to data originators and recipients. The meanings of composite properties were made precise by standardising the order in which individual properties are enforced. The operators defined over them allowed overall security requirements along the path to be expressed as a function of security requirements of interacting parties. By defining all security properties in an unambiguous way many of the past security flaws caused by multiple interpretations of common security properties can be avoided. The hierarchical structure made it possible to specify exact security requirements, thus avoiding overheads caused by over-prescription of security requirements. Standardizing the meaning of fine-grained security properties allow security requirements of interacting entities to be specified explicitly. Past matchmaking techniques expressed possible transitions using commitments or permitted workflow as the basis but did not include security requirements for permitted interactions. The security levels in this model with precise meanings attached to them allow such transitions to be annotated with the necessary security levels.

Provably Secure Schemes Providing Fine-grained Security Properties

A new approach allowed e-commerce protocols meeting end-to-end security requirements similar to SET protocols to be synthesised at runtime by combining provably correct fine-grained security schemes. Though a number of previous attempts have been made to synthesise protocols using standard challenge-response mechanisms or composition of protocols, no past attempt was made to combine these techniques to form fine-grained security schemes. Furthermore, the long elapsed times required for previous synthesis approaches made them unsuitable for service composition. Axioms involving protocol actions were used to prove the basic challenge-response schemes devised, while invariants enforced by the composition logic guarantee the validity of composite schemes. It was also shown that these schemes can be further strengthened to resist common attacks such as type flaw and replay attacks.

Cost Model for Synthesised Protocols

One novel aspect of the proposed synthesis approach was that it combined formal methods used for security with performance measures. The bandwidth and computational costs of synthesised protocols

were expressed in terms of underlying security strength allowing selection of protocols with better security performance trade-offs. Pervasive use of security protocols across various resource constrained devices and bandwidth constrained networks make such trade-offs inevitable. Furthermore, by expressing the cost of security in terms of security strength allowed security strength to be lowered to meet performance constraints.

Interleaving Technique for Extending Two-Party Schemes

A novel interleaving technique devised allowed any two-party scheme to be extended to any number of recipients at runtime. Such schemes allow data to be delivered in a specific order by piggybacking the necessary cryptographic elements. The validity of extended schemes was proved using Strand Spaces. The interleaved schemes provide a number of benefits. First, direct evidence from originator and recipients are included even though data is sent through third parties. Second, the number of protocol steps required at data source is significantly reduced when data must be despatched to multiple recipients with various security properties.

Framework for Institutional Trust

A new institutional trust framework devised for e-commerce can help overcome the perception of risk involved in trading with unknown e-commerce entities. This framework allowed key messages to be passed through trusted, category specific endorsement intermediaries. The endorsement intermediary network consisted of nodes representing trading entities and authorised intermediaries, while edges represented the extent of trust relationships between nodes. All intermediaries were made accountable for their endorsements. The path endorsement trust proposed gave a measure of indemnity based on the number of endorsement intermediaries and the extent of the trust relationships along the path. By combining category specific trust and endorsements with security mechanisms for detecting trust breaches, the proposed framework provided a much stronger basis for trust propagation than the transitive trust models proposed in the past. Large networks were decomposed into hierarchical domains using clustering techniques. Simulation results showed average search time for trusted path through domains with medium trust coupling grows linearly. Based on these results and the hierarchical clustering techniques devised it was estimated that trusted paths can be retrieved within one second, even for networks of up to 800,000 nodes. Offline generation of trusted paths can improve performance even further.

Self-regulating Centralised Trust Networks

The trust network was centralized by modelling trust as a derived value based on past conduct, trust disposition and trusting beliefs. The network was also made self-regulating using trust transfer and trust evolution policies. Trust evolution policies either raised or lowered trust relationships based on transaction outcomes. The extent of these changes was made to reflect trusting beliefs in underlying domains for the data category. Trust transfer policies allowed new trust relationships to be established

reflecting their own trust disposition. Simulation results showed that the trust transfer threshold should be varied over time if entities are to build and preserve their trust capital. These simulation results demonstrate how optimal trust disposition can be made a function of trusting beliefs in the domain by combining trust transfer and trust evolution policies.

Deriving End-to-End Accountability Schemes

End-to-end accountability schemes were derived using the notion of proof obligations. End-to-end accountability schemes for security properties providing assurances to recipients (A,DI,TB) were derived by transferring proof obligations through intermediaries until they were directly discharged using evidence from the data originator. Similarly end-to-end accountability schemes for security properties providing assurances to data source (RNR) were derived by delegating proof obligations through intermediaries until direct evidence were obtained from all final recipients. End-to-end assurances follow naturally when all proof obligations are discharged either directly or indirectly. Any trust breach or invalid endorsement can be narrowed to a specific intermediary or trading entity that fails to discharge its proof obligations. Past research with accountability was mainly restricted to verifying whether existing protocols can make participants accountable for their actions.

Enforcing End-to-End Security

An SSES scheme devised allows the server signed metadata, consisting of data hash, data category, trusted path and required security levels along the path, to act as a form of protocol description. Each intermediary and recipient along the trusted path enforces these security levels using the proven two party schemes in their possession. Such a scheme allowed end-to-end security and trust requirements to be met.

6.2 Future Work

Future work could extend the research undertaken in this thesis in a number of different directions. Some of the possible areas for research are suggested below.

Endorsement Functions

The current model does not include implementation details of endorsement functions. More work must be carried out about implementation issues dealing with interfaces, quality of service, maintenance and access control.

Trust Growth

Currently, trust growth for successful transactions does not take into account the value of transactions. The model can be extended to set rewards and penalties as a function of transaction value. Currently trust transfer occurs only when trust relationships exist with trusted neighbours. Trust transfer can be extended to consider transitive trust relationships modelling indirect referral in traditional commerce.

Fine-Grained Properties and Schemes

The security properties can be extended to include other recent properties in e-commerce such as anonymity and privacy. Current two-party schemes can incorporate techniques to resist attacks such as replay and type-flaw. These can be further strengthened to resist other forms of attacks including reflection, oracle and algebraic attacks. Specialized model checkers can be devised to verify whether the composed schemes continue to meet the original requirements. Currently composed schemes use distinct encryption techniques combining signing with encryption. The overheads for separate signing and encryption can be reduced by devising schemes based on signcryption which combines signing and encryption into a single operation.

Synthesising Security Protocols for E-Commerce Workflow

The security schemes devised in this thesis can be extended to e-commerce workflow patterns. This requires identifying common patterns including non-deterministic ones and identifying how security levels can be aggregated over them. If a knowledge based approach is used, it requires modelling the initial world and all possible final worlds.

Varying Security Strength

The current synthesis framework requires the same security strength to be used throughout the protocol. Better security/cost trade-offs are possible if a flexible synthesis framework allows security strength to be varied within the same protocol, reflecting the value of data, the type of entities exchanging the data and the level of threats posed. Synthesising such protocols require the type of entities and their performance/security requirements to be explicitly represented.

Glossary

Correspondence Properties

Properties that specify temporal relationships between the various events in the protocols such as a send event in one entity and a receive event in another.

Combinatorial Explosion

Exponential growth in the number of possible combinations.

Time Stamp

A time stamp is the recorded time of an event. The time stamp mechanism is used for a wide variety of synchronisation purposes. Time stamps are common in security protocols including Kerebos.

Logical Time Stamp

Logical time stamp increments a scalar value each time a send/receive event occurs using a logical clock. Logical clocks allow data recipients to detect replayed attacks through repeated logical clock values and are less costly than maintaining synchronised clocks.

Fair Exchange Protocol

A fair exchange protocol ensures that no party gains any advantage by prematurely terminating a protocol.

One-Time Key

With one-time key, a private key is used only once to encrypt data which is then decrypted using a matching key. One-time keys cannot be identified by analysing successive messages as they are used only once.

Spanning Tree

A spanning tree of a graph is a sub-graph which is a tree (no cycles) that connects all the vertices in the graph.

Symmetric Group Key

Symmetric group keys are identical keys shared by a group to enforce confidentiality.

1. Yan, Y. and X. Zheng, *A Planning Graph Based Algorithm for Semantic Web Service Composition* in *IEEE Conference on E-Commerce Technology* 2008: pages 339-342.
2. Singaravelu, L. and C. Pu, *Fine Grain, End-to-End Security for Web Sevice Compositions* in *IEEE International Conference on Services Computing* 2007: pages 212-219.
3. Anderson, B.B., et al., *The application of Model Checking for Securing e-Commerce Transactions in Communications of the ACM* 49(6) pages 48-53 (2006).
4. Antoniou, G., L. Batten, and U. Parampalli, *A Trusted Approch to E-Commerce* in *Secure Data Management 2008:* pages 119-132, Auckland, New Zealand.
5. Antonio, J. and J. Zhou, *Intermediary Non-repudiation Protocols* in *IEEE International conference on E-Commerce* 2003: pages 207-214, Newport Beach, CA, USA.
6. Bella, G., F. Massaci, and L.C. Paulson, *Verifying the Set Purchase Protocols* in Journal of Automated Reasoning 36: pages 5-37 (2006).
7. Zhou, J. and D. Gollmann, *An efficient Non-repudiation Protocol* in *Computer Security Foundation Workshop* 1997: pages 126-132, Rockport, Massachusetts.
8. Chun, O. and B. Jonathan, *An improved formal specification of the Internet Open Trading Protocol* in *ACM Symposium on Applied Computing* 2004: pages 779-783, Nicosia, Cyprus.
9. Sherif, M.H., *Protocols for Secure Electronic Commerce*, Second Edition, 2004, New Jersey: CRC Press, ISBN 0-8493-1509-3.
10. Hanaoka, G., Y. Zheng, and H. Imai, *LITESET: A Light-Weight Secure Electronic Transaction Protocol* in *Third Australasian Conference on Information Security and Privacy* 1998: pages 215-226, Australia.
11. Paulson, L., *Verifying the SET protocol: Overview* in *FASec* 2002: pages 4-14, London.
12. Ruiz, M.C., et al., *Analysis of the SET E-commerce Protocol Using a True Concurrency Process Algebra* in *Symposium on Applied Computing* 2006: pages 879-886.
13. Seo, M. and K. Kim, *Another Improved SET: Slim SET* in *Proceedings of the Symposium on Cryptography and Information Security* 2000, Okinawa, Japan.
14. Zhou, H. and S.N. Foley, *A Collaborative Approach to Autonomic Security Protocols* in *New Security Paradigms Workshop* 2004: pages 13-21, Nova Scotia, Canada: ACM.
15. Carminati, B., E. Ferrari, and P.C.K. Hung, *Web Service Composition: A Security Perspective* in *International Workshop on Challenges* in *Web Information Retrieval and Integration* 2005: pages 248-253.
16. Singh, M.P., *Trustworthy Service Composition: Challenges and Research Questions* in *International Conference on Autonomous Agents and Multiagent System* 2002: pages 39-52, Bologna, Italy.
17. Yang, S.J.H., et al., *Composition and Evaluation of Trustworthy Web Services* in *International Journal of Web and Grid Services* 2(1) pages 5-24 (2006).
18. Lowe, G., *An Attack on the Needham-Schroeder Public-Key Authentication Protocol* in *Information Processing Letters* 1995. 56(3) pages 131-133.
19. Guttman, J.D., et al., *Programming Cryptographic Protocols* in *Trustworthy Global Computing* in *International Symposium* 2005: pages 116-145.
20. Buttyan, L., S. Staamann, and U. Wilhelm, *A simple logic for authentication protocol design* in *Computer Security Foundations Workshop* 1998: pages 153-162.
21. Woo, T.Y.C. and S.S. Lam. *A lesson on authenticated protocol design* in *ACM Operating Systems Review* 28(3) pages 24-37 (1994).
22. Wang, W., et al., *E-Process Design and Assurance Using Model Checking* in *IEEE Computer* 33(10): pages 48-53 (1999).
23. Hongbin Zhou, Simon N. Foley, *Fast automatic security protocol generation* in *Journal of Computer Security* 20 (2-3): pages 119-167 (2012).
24. Hao, C., J.A. Clark, and J.L. Jacob, *Automated Design of Security Protocols* in *Computational Intelligence* 20(3): pages 503-516 (2004).
25. Anupam Datta, Ante Derek, John C. Mitchell, Dusko Pavlovic, *A derivation system and compositional logic for security protocols* in *Journal of Computer Security* 13(3): pages 423-482 (2005).
26. Anupam Datta, Ante Derek, John C. Mitchell, Arnab Roy, *Protocol Composition Logic (PCL)* in *Electronic Notes on Theoretical Computer Science* vol 172: pages 311-358 (2007).
27. Kearney, P., *Trust and Security in Virtual Organizations in BT Technology Journal* 24(2) pages 209-213 (2006).

28. Anderson, P. and E. Anderson, *The New E-Commerce Intermediaries in MIT Sloan Mangement Review* 43(4) (2002).
29. Dikaiakos, M.D., *Intermediary Infrastructure for the World Wide Web* in *Computer Networks*, vol 45: pages 421-447 (2004).
30. Kailar, R., *Accountability in Electronic Commerce Protocols* in *IEEE Transactions on Software Engineering*, 22 (5): pages 313-328 (1996).
31. Meadows, C. *What Makes a Cryptographic Protocol Secure? The Evolution of Requirements Specification* in *Formal Cryptographic Protocol Analysis* in *ESOP* 2003: pages 10-21.
32. Paul, S. and M. Catherine, *A Formal Language for Cryptographic Protocol Requirements* in *Codes and Cryptography*. 7(1-2): pages 27-59 (1996).
33. Lowe, G., *A hierarchy of authentication specifications* in *10th Computer Security Foundations Workshop*. 1997: pages 31-44.
34. Pearce, C., P. Bertok, and C. Thevathayan, *A Protocol for Secrecy and Authentication within Proxy-Based SPKI/SDSI Mobile Networks* in *Asia Pacific Information Technology Security Conference* 2004: Gold Coast, Australia.
35. Rosenberg, F., et al., *An End-to-End Approach for Qos Aware Service Composition* in *IEEE International Enterprise Distributed Object Computing Conference*. 2009, Washington.
36. Chalouf, M.A., et al., *On Tightly Managing End-toEnd QOS and Security for IPTV Service delivery* in *IWCMC* 2009, ACM: Leipzig.
37. Ketchpel, S.P. and H. Garcia-Molina., *Making Trust Explicit* in *Distributed Commerce Transactions*, 1996: pages 270-281, Hong Kong.
38. Melideo, G. and G. Proietti, *Truthful Mechanism for Building Trust* in *E-Commerce. Certification and Security in Inter-Organizational E-Services* 2005: pages 101-112.
39. Olsson, O., *Trust in eCommerce - the ontological status of trust* in *ECOM-02*. 2002, Gdansk.
40. Su, J. and D. Manchala, *Building Trust for Distributed Commerce Transactions* in *17th International Conference on Distributed Computing Systems* 1997.
41. Sun, Z., et al., *Experience-Based Trust* in *E-Commerce* in *IFIP International Federation for Information Processing* 2007: pages 643-651.
42. Wantang, J., *E-Commerce Trust Building under the involvement of Intermediary Mechanism* in *International Forum on Computer Science Technology and Applicatiions* 2009: pages 388-391.
43. Neuman, B.C., *Security, Payment, and Privacy for Network Commerce* in *IEEE Journal on Selected Areas in Communications* 13(8): pages 1523-1531 (1995).
44. Andert, D., R. Wakefield, and J. Weise, *Trust Modeling for Security Architecture Development* in *Sun MicroSystems BluePrints (2002)*.
45. Sandhu, R., *Good-Enough Security in IEEE Internet Computing* 4(1), 2003.
46. Prasithsangaree, P. and P. Krishnamurthy, *On a framework for energy-efficient security protocols in wireless networks* in *Computer Communications* 27(17): pages 1716-1729.
47. Guttman, J.D., *Security Protocol Design via Authentication Tests* in *CSFW'02*. 2001.
48. Stubblebine, S.G., *Recent-Secure Authentication* in *IEEE Symposium on Research* in *Security and Privacy* 1995: pages 224-235.
49. Menasce, D.A., *Security Performance* in *IEEE Internet Computing* 7(3) pages 84-87 (2003).
50. Kambourakis, G., A. Rouskas, and S. Gritzalis, *Performance evaluation of public key-based authentication* in *Future Mobile Communication Systems* in *EURASIP Journal on Wireless Communications and Networking* (1): pages 184-197 (2004).
51. Miller, S.K., *Facing the Challenge of Wireless Security* in *IEEE Computer* vol 34: pages 16-18 (2001)
52. Ravi, S., Ragunathan, A., Potlapally, N., Sankaradass M., *System design methodologies for a wireless security processing platform* in *Annual ACM IEEE Design Automation Conference* 2002: pages 777-782, Louisiana, USA.
53. Lin, C. and V. Varadharajan, *Trust Based Risk Management for Distributed System Security - A New Approach* in *Availability* in *IEEE conferenece on Reliability and Security*. 2006, pages 6-13.
54. Josang, A., E. Gray, and M. Kinateder, *Simplification and Analysis of Transitive Trust Networks* in *Web Intelligence and Agent Systems* 4(2): pages 139-161 (2006).
55. Marco Cadoli and Tony Mancini, *Automated Reformulation of Specifications by Safe Delay of Constraints* in *Journal of Artificial Intelligence* 170(8) pages 779-801 (2006).

56. R. Gotzhein and Bochmann, G.V., *Deriving Protocol Specifications from Service Specifications including Parameters* in *ACM Transactions on Computer Systems,8(4)*: pages 255-283 (1990).
57. Probert, R.L. and K. Salech, *Synthesis of Communication Protocols: Survey and Assessment* in *IEEE Transactions on Computers* **40**(4): pages 468-476 (1991).
58. Chua, P.-Y. and M. Liu., *Protocol Synthesis* in *State Transition Model* in *Computer Software and Applications Conference* 1998, pages 505-51.
59. Meyden, R.v.d. and M.Y. Vardi, *Synthesis of Distributed Systems from Knowledge based Specifications* in *Concurrency Theory*, 2005, pages 562-576.
60. Saïdi, H., *Towards automatic synthesis of security protocols* in *Proceedings of the Logic-Based Program Synthesis Workshop, AAAI Spring Symposium* 2002, Stanford University, California.
61. Datta, A., J. Mitchell, and A. Derek, *Secure Protocol Composition* in *ACM workshop on Formal methods in security engineering* 2004: pages 11-23.
62. Cheng, Z., M.P. Singh, and M.A. Vouk, *Composition Constraints for Semantic Web Services* in *International Workshop on Real World RDF and Semantic Web Applications*. 2002.
63. Milanovic, N. and M. Malek, *Current Solutions for Web Service Composition* in *IEEE Internet Computing*, 2004. **8** (6): pages 51 - 59.
64. Chiu, D.K.W., et al., *Workflow View Driven Cross-Organizational Interoperability in a Web-Sevice Environment* in *WES 2002*: pages 41-56.
65. Kim, K., J.A. Abraham, and J. Bhadra, *Model Checking of Security Protocols with Pre-configuration* in WISA 2003: 1-15, Jeju Island, Korea.
66. Milner, R., J. Parrow, and D. Walker, *A Calculus of Mobile Processes, Parts I and II* in *Information and Computation* **100**: pages 1-77 (1992).
67. Shaikh, S. and V. Bush, *Analysing the Woo-Lam Protocol Using CSP and Rank Functions* in *Journal of Research and Practice in Information Technology*, 2006. **38**(1): pages 19-29.
68. Gramlich, B., *Strategic Issues, Problems and Challenges in Inductive Theorem Proving* in *5th International Workshop on Strategies in Automated Deduction*, 2004.
69. Schneider, S.A., et al., *Modelling and Analysis of Security Protocols*, 2001, Addison Wesley, ISBN-13: 978-0-201-67471-2.
70. Abadi, M. and A.D. Gordon, *A calculus for cryptographic protocols* in *Information and Computation* 1999. **148**(1): pages 1-70.
71. IBM, Microsoft, and Verisign, *Web Service security (WS-Security)*. 2002.
72. O'Neil, M., *Web Services Security*. 2003: McGraw Hill, ISBN-13: 978-0072224719.
73. Perrig, A., and Song, D., *A First Step towards the Automatic Generation of Security Protocols* in *Network and Distributed System Security Symposium* 2000: pages 73-83.
74. Song, D., Berezin, S., and Perrig, A., *Athena: A Novel Approach to Efficient Automatic Security Protocol Analysis* in *Journal of Computer Security*, 2001, 9(1): pages 47-74.
75. Crazzolara, F. and G. Winskel, *Composing Strand Spaces* in *Foundations of Software Technology and Theoretical Computer Science* 2002, Springer-Verlag, Kanpur.
76. Guttman, J.D. and F.J. Thayer, *Protocol independence through disjoint encryption* in *Computer Security Foundations Workshop* 2000: pages 24-34. Cambridge, England.
77. Hoare, C.A.R., *An axiomatic basis for computer programming* in *Communications of the ACM* **12**(10): pages 576-583 (1969).
78. Durgin, N., J. Mitchell, and D. Pavlovic, *A compositional logic for proving security properties of protocols* in *Journal of Computer Security* **11**: pages 677 - 721 (2003).
79. Needham, R.A.a.R., *Robustness principles for public key protocols* in *Journal of the ACM* (JACM) **46**(5) (1995).
80. Heather, J., G. Lowe, and S. Schneider, *How to prevent type flaw attacks on security protocols in Journal of Computer Security* **11**(2): pages 217-244 (2003).
81. Yafen Li, W.Y. and C.-W. Huang, *On Preventing Type Flaw Attacks on Security Protocols With a Simplified Tagging Scheme* in *Journal of Information Science and Engineering* **21(1)**: pages 59-84 (2005).
82. Malladi, S., J. Alves-Foss, and R. Heckendorn, *On preventing replay attacks on security protocols* in *Proc. ICSM '02 (2002)*.
83. Syverson, P. *Taxonomy of Replay Attacks* in Proceedings of the 7th IEEE *Computer Security Foundations Workshop*, 1994: pages 187-191.

84. Tuomas, A., *Strategies against replay attacks* in *In Proceedings of the 10th IEEE Computer Security Foundations Workshop*, 1997.
85. Schneier, B., *Applied Cryptography*, second edition, 1996, John Wiley, ISBN-13: 978-0471117094.
86. Challal, Y., A. Bouabdallah, and H. Bettahar, *H2A: Hybrid Hash-chaining scheme for Adaptive multicast source authentication of media-streaming* in *Computers and Security*, **24**(1): pages 57-68 (2005).
87. Abadi, M. and R. Needham, *Prudent Engineering Practice for Cryptographic Protocols* in IEEE Transactions on Software Engineering **22**(1): pages 6-15 (1996).
88. Anderson, R. and R. Needham, *Programming satan's computer* in *Computer Science Today: Recent Trends and Developments*. 1995: pages 426-440.
89. Cheung, T. and S. Chanson, *Design and Implementation of a PKI-Based End-to-End Secure Infrastructure for Mobile E-Commerce* in *Second International Conference on Web Information Systems Engineering (WISE'01)*. 2001. Los Alamitos, CA, USA.
90. Tak, S.W., et al., *Design and Evaluation of adaptive secure protocol for E-Commerce* in *Tenth Conference on Computer Communications and Networks*. 2001, pages 32-39.
91. Tak, S.W., Y. Lee, and E.K. Park, *A Software Framework for Non-repudiation Service* in *Electronic Commerce based on the Internet* in *Eleventh Conference on Computer Communications and Networks*, 2002, pages 182-189.
92. Yi, S., P. Naldurg, and R. Kravets, *A Security-Aware Routing Protocol for Wireless Ad Hoc Networks* in *ACM Symposium on Mobile Ad Hoc Networking & Computing* 2001, pages 299-302.
93. Hinton, H.M., *Under-Specification, Composition and Emergent Properties* in *New Security Paradigms Workshop*, 1997, pages 83-93, Langdale, UK.
94. Yolum, P. and M. Singh, *Commitment Machines* in *International Workshop on Intelligent Agents* 2001: pages 235-247.
95. Wombacher, A., P., Fankhauser, and E. Neuhold, *Matchmaking for Business Process Based on Choreographies* in *IEEE Conference on e-Technology, e-Commerce and e-Services*, 2005: pages 359-368.
96. Boyd, C., *Towards extensional goals in authentication protocols* in *DIMACS workshop in Design and Formal Verification of Security Protocols*, 1997, New Jersey.
97. Schneider, S., *Veifying Authentication Protocols in CSP* in *Transactions on Software Engineering* **24**(9) pages 741-758 (1998).
98. Denning, D.E., *A lattice model of secure information flow* in *Communications of the ACM* **19**(5): p. 236-243 (1976).
99. Myers, A.C. and B. Liskov, *Complete Safe Information Flow with Decentralized Labels* in *IEEE Symposium on Security and Privacy* 1998: pages 186-197, Oakland, California.
100. Abadi, M., *Secrecy by typing in security protocols* in *Journal of the ACM* **46**(5): pages 749-786 (1999).
101. Boyd, C., *Some Applications of Multiple Key Ciphers* in *EUROCRYPT Workshop* 1988: pages 455-467.
102. Dolev, D. and A.C. Yao., *On the security of public key protocols* in *IEEE Transactions on Information Theory* **29**(2) (1983).
103. Adrian, P., Dawn Son, and J.D. Tygar, *ELK, a New Protocol for Efficient Large-Group Key Distribution* in *IEEE Symposium on Security and Privacy*, 2001.
104. Menezes, A., P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, 1997, CRC Press, ISBN:0849385237.
105. Ray, I. and I. Ray, *Fair exchange in E-commerce* in *ACM SIGecom Exchanges* 2002: **3**(2) pages 9-17.
106. Fabrega, F.J.T. and J.C. Herzog, *Why is a Security Protocol Correct?* in *IEEE Symposium on Security and Privacy* 1998: pages 160-171.
107. Chung, H. and C. Neuman, *Modelling the Relative Strength of Security Protocols* in *ACM Workshop on Quality of protection* 2006: pages 45-48.
108. Miltchev, S.M., S. Ioannidis, and A.D. Keromytis, *A Study of the Relative Costs of Network Security Protocols* in *USENIX Annual Technical Conference* 2002: pages 41-48.
109. Cremers, C., *Compositionality of Security Protocols: A research Agenda* in *Electronic Notes in Theoretical Computer Science* 2006: pages 99-110.

110. Potlapally, N.R., et al. *Analyzing the Energy Consumption of Security Protocols* in *International Symposium on Low Power Electronics and Design* 2003: pages 30-35.
111. Blaze, M., J. Feigenbaum, and K. Lacy, *Decentralized Trust Management* in *IEEE Symposium on Security and Privacy 1996:* pages 164-173.
112. Li, H. and M. Singhal, *Trust Management in Distributed Systems* in *IEEE Computer Society* 2007: 40(2) pages 45-53.
113. McKnight, D.H., V. Choudhury, and C. Kacmar, *Developing and Validating Trust Measures for e-Commerce: An Integrative Typology* in *Information Systems Research*: **13**(3) pages 334-359 (2002).
114. Antoniou, G. and L. Batten, *E-commerce: Protecting Purchaser Privacy to Enforce Trust* in *Electronic Commerce Research and Applications* **11**(4) pages 421-456 (2011).
115. Katsikas, S.K., J. Lopez, and G. Pernul, *Trust, Privacy and Security in E-Business: Requirements and Solutions* in *10th Panhellenic Conference on Advances in Informatics* 2005: pages 548-558.
116. Koufaris, M. and W. Hampton-Sosa, *The development of initial trust in an online company by new customers* in I*nformation and Management* **41**(3) pages 377-397 (2004).
117. Serva, M.A., J. Benamati, and M.A. Fuller, *Trustworthiness in B2C E-Commerce: An Examination of Alternative Models* in *ACM SIGMIS Database* **36**(3) pages 89-108 (2005).
118. Mayer, R.C., J.H. Davis, and F.D., Schoorman, *An Integrative Model of Organizational Trust* in *Academy of Management Review* **20**(3): pages 709-734 (1995).
119. Josang, A., S. Hird, and E. Faccer, *Simulating the Effect of Reputation Systems on e-Markets* in *International Conference on Trust Management* 2003: pages 179-194.
120. Dong, C., G. Russello, and N. Dulay, *Trust Transfer in Distributed Systems* in *IFIP International Federation for Information Processing* 2007: pages 17-29, New Brunswick, Canada.
121. Lai, C., G. Medvinsky, and B.C. Neuman, *Endorsements, Licensing, and Insurance for Distributed System Services* in *Conference on Computer and Communications Security* 1994: pages 170-175.
122. Resnick, P. and R. Sami, *Sybilproof Transitive Trust Protocols* in *ACM Conference on Electronic Commerce* 2009: pages 345-354, Stanford.
123. Srinivasan, S, *Role of Trust in E-business Success* in *Information Management & Computer Security*, **12**(1): pages 66-72 (2004).
124. Siyal, M.Y. and M. Barkat, *A Novel Trust Service Provider for Internet Based Commerce Applications* in *Internet Research*, **12**(1): pages 55-65 (2002).
125. Yolum, P. and M.P. Singh, *Engineering Self-Organizing Referral Networks for Trustworthy Service Selection* in *IEEE Transactions on Systems, Man, and Cybernetics* **35**(3) pages 17-29, (2007).
126. Maximilien, E.M. and M.P. Singh, *Reputation and Endorsement for Web Services* in *SIGecom Exchanges* 3(1) pages 24-31 (2002).
127. Pan, J., et al., *A Dynamic Trust Network Based Simulation Framework for Reputation-based Service Selection* in *Internetware* 2009, Beijing, China.
128. Atif, Y., *Building Trust in E-Commerce* in *IEEE Internet Computing* 2002: 6(1) pages 18-24.
129. Chen, Y., *Max-Minimum Algorithm for Trust Transitivity in Trustworthy Networks* in *International conference on Web Intelligence and Intelligent Agent Technology* 2009: (3) pages 62-64.
130. Guha, R., et al., *Propagation of Trust and Mistrust* in *13th international conference on World Wide Web* 2004: pages 403-412.
131. Christianson, B. and W.S. Harbison, *Why Isn't Trust Transitive?* in *Security Protocols International workshop* 1996: pages 171-176.
132. Josang, A. and S. Pope, *Semantic Constraints for Trust Transitivity* in *Asia-pacific Conference on Conceptual Modelling* 2005: vol 43 pages 59-68.
133. Liu, G., Y. Wang, and M. Orgun, *Optimal Social Trust Path Selection in Complex Social Networks* in *IEEE International Conference on Web Services* 2011: pages 41-48.
134. S. L. Yong-Yeol Ahn, J.P.B, *Link Communities Reveal Multiscale Complexity in Networks.* Nature 2010: vol 46 pages 761-764.

135. Li, M. and A. Bonti, *Trust Evaluation for Indirectly Connected Nodes using Link Community Information* in *First NSS Workshop on Mobile and Online Social Networks*. 2011, IEEE: Milan. pages 19-26.
136. Liu, G., Y. Wang, and M. Orgun, *Trust Transitivity in Complex Social Networks* in *AAAI Conference in Artificial Intelligence*, 2011, San Francisco.
137. Campadello, S., *The Green Card Protocol: An Identification Protocol for Decentralized Systems* in *International Symposium on a World of Wireless, Mobile and Multimedia* 2006: pages 647-651.
138. Ali, A.S., A. Ludwig, and O.F. Rana, *A Cognitive Trust-Based Approach for Web Service Discovery and Selection* in *Third European Conference on Web Services* 2005, Washington DC, USA.
139. Manchala, D.W., *E-Commerce Trust Metrics and Models* in *IEEE Internet Computing* 4(2) pages 46-34 (2000).
140. Sedgewick, R., *Algorithms in C - Graph Algorithms*, third edition, 2001, Addison Wesley, ISBN-10: 0201316633**.**
141. Djatmiko, M., et al., *Trust-based Content Distribution for Mobile Ad Hoc Networks* in *IEEE International Symposium on Modelling* 2011: pages 433-436.
142. Karlan, D., Markus Mobius, T. Rosenblat, and Adam Szeidl., *Trust and Social Collateral* in *Quarterly Journal of Economics*, 2009: **124**(3) pages 1307-1361.
143. Ahn, Y.-Y., S. Han, and H. Kwak, *Analysis of Topological Characteristics of Huge Online Social Networking Services* in *International World Wide Web Conference* 2007: pages 835-844.
144. Suzuki, T., et al., *A system for end-to-end Authentication of Adaptive Multimedia Content* in *Conference on Communications and Multimedia Security* 2004: pages 237-249.
145. Srinivas, B.S. and T. Chan, *Security Threats and Risks with Content Transformation Intermediaries* in *10th International Conference on Telecommunications* 2003: pages 608-613.
146. Koglin, Y., D. Yao, and E. Bertino, *Efficient and Secure Content Processing and Distribution by Cooperative Intermediaries* in *IEEE Transactions on Parallel and Distributed Systems* 2008. **19**(5): pages 615-626.
147. Dikaokos, M, *Intermediaries for the World-Wide-Web Overview and Classification* in *Seventh IEEE International Symposium on Computers and Communicatiosn* 2002: pages 231-236.
148. Gentry, C., et al., *End-to-End Security in the Presence of Intelligent Data Adapting Proxies. IEEE Journal on Selected Areas in Communications* 2005: **23**(2) pages 464-473.
149. Stubblebine, S.G. and R.N. Wright, *An Authentication Logic with Formal Semantics supporting Synchronization, Revocation and Recency* in *IEEE Transactions on Software Engineering* **28**(3) pages 256-285 (2002).
150. Kremer, S. and Markowitch, O., *A multi-party non-repudiation protocol* in *Third International Conference on Security and Cryptography* 2000: pages 271-280.
151. Caronni, G., *Walking the Web of Trust* in *WETICE 2000*: pages 153-158.
152. Klos, T.B., *Trusted Intermediaries Agents in Electronic Trade Networks* in *AAMAS* 2005: pages 1249-1250.
153. Crispo, B., *Delegation Protocols for Electronic Commerce* in *6th IEEE Symposium on Computers and Communications* 2001: pages 674-679.
154. Xiao, Y., *Security in Distributed, GRID, Mobile and Pervasive Computing*, 2007, Auerbach, ISBN 9780849379215.
155. Crispo, B. and G. Ruffo, *Reasoning about Accountability within Delegation* in *Third International Conference on Information and Communications Security 2001*: pages 251-260.
156. Parikh, R. and R. Ramanujam, *A Knowledge Based Semantics of Messages.* Journal of Logic, Language and Information archive, 2003. **Volume 12** (Issue 4 (Fall 2003) table of contents): p. Pages: 453 - 467
157. Huang, A.-C. and P. Steenkiste, *Building Self-configuring Services Using Service-specific Knowledge*, in *High Performance Distributed Computing* 2004.
158. Meyden, T.V.d. and K. Su. *Symolic model checking the knowledge of the dining cryptographers*. in *Computer Security Foundation Workshop*. 2004. IEEE.
159. Meyden, R.v.d. and M.Y. Vardi, *Synthesis from Knowledge based Specifications*, in *Concurrency Theory* 2005.

160. Haibin Zhang., et al., *Efficient Contextual Trust Computation* in *11th International Conference on Trust, Security and Privacy in Computing and Communications*.
161. Punam Bedi., et al., *Trustworthy Service Provider Selection in Cloud Computing Environment* in *2012 International Conference on Communication Systems and Network Technologies*.
162. Y. Zhong, Y. Lu, and B. Bhargava., An authorization framework based on uncertain evidence and dynamic trust.,Technical Report, CSD-TR 04-009, Purdue University, 2004.
163. Ahmet Burak Can, Bharat Bhargava., *SORT: A Self-Organizing Trust Model for Peer-to-Peer Systems*. IEEE Transactions on Dependable and Secure Computing, 2013. **Volume 10** (1) Pages: 14-27
164. X. Li, Z. Jia, P. Zhang and H. Wang., *Trust-based On-demand Multipath routing in mobile adhoc networks*. IET Information Security, 2010, Volume 4(4): Pages 212-232.
165. Michael Huth, Mark Ryan., *Logic in Computer Science*, second edition, 2004, Cambridge Univers trustity Press, ISBN-10: 05254310X**.**
166. M. Burrows, M. Abadi, and R. Needham., *A Logic of Authentication.,* in *Proceeding of the Royal Society*, 1989, Volume A426 Pages 233-271.
167. Chares Thevathayan, James Harland, Peter Bertok., *Endorsement Trust Model* in *Proceedings of 12th International Conference on Trust, Security and Privacy in Computing and Communications,* 2013.
168. Mehdi Azarmi, Bharat Bhargava,Pelin Angin, Rohit Ranchal, Norman Ahmed, Asher Sinclair, Mark Linderman., *An End-to-End Security Auditing Approach for Service Oriented Architectures* in the proceedings of 31st *International Symposium on Reliable Distributed Systems,* 2012.
169. Oluwafemi Ajayi, Richard Sinnott and Anthony Stell., *Towards Decentralised Security Policies for e-Health Collaborations,* in the proceedings of 2nd *International Conference on Emerging Security Information, Systems and Technologies,* 2008.
170. Priyanka Dadhich, Kamlesh Dutta,M.C. Govil., *On the Approach of Combining Trust and Security for Securing Mobile Agents:Trust Enhanced Security* in *Proceedings of International Conference on Computer and Communication Technology*, 2013.